Similarity Search and Data Mining Techniques for Advanced Database Systems.

Dissertation im Fach Informatik
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig-Maximilians-Universität München

von

Alexey Pryakhin

Tag der Einreichung: 24.11.2006
Tag der mündlichen Prüfung: 21.12.2006

Berichterstatter:
Prof. Dr. Hans-Peter Kriegel, Ludwig-Maximilians-Universität München
Prof. Dr. Daniel A. Keim, Universität Konstanz

Acknowledgement

I would like to express my warmest gratitude to all the people who supported me during the past three years while I have been working on this thesis. I avail myself of the opportunity to thank them, even if I cannot mention all of their names here.

First of all, I would like to express my warmest and sincerest thanks to my supervisor, Professor Dr. Hans-Peter Kriegel, who provided the productive and inspiring environment and created a great working atmosphere within our group. I warmly thank Professor Dr. Daniel Keim for his immediate willingness to act as a second referee for my thesis.

This work could not have grown and matured without the discussions with my colleagues in the database research group. In particular, I would like to give my thanks to Elke Achtert, Johannes Aßfalg, Karsten Borgwardt, Professor Dr. Christian Böhm, Stefan Brecheisen, Dr. Karin Kailing, Dr. Peer Kröger, Peter Kunath, Dr. Matthias Schubert, Matthias Renz, Arthur Zimek for their help, support, interesting hints, constructive and productive teamwork. Furthermore, I want to thank Alexander Harhurin, Otmar Hilliges, Florian Vorberger for other fruitful multidisciplinary discussions about software engineering, similarity of multimedia objects, and music genres which were useful for this work. Last, but not least, I had the pleasure to supervise and to work with several students who supported my work and who have been beneficial for this work. In particular, I would like to mention here Oleg Galimov, Franz Graf, Michael Gruber, Georg Straub, Michael Kats, Sergey Wetzstein, Andrew Zherdin, and Karina Zöhrer.

I would like to express my deep appreciations to Susanne Grienberger. Besides bearing much of the administrative burdens for me, she helped me a lot by carefully reading the thesis, and by polishing the English. I wish to specially thank Franz Krojer for taking care of our hard- and software environment and for his invaluable technical hints that allowed me to save a lot of time during experimental evaluation.

I owe special thanks to my wife Anna for her love, care, and patience during the period of my PhD thesis. Without her encouragement and understanding, it would have been impossible for me to complete this work. I would also like to thank the rest of my family and my friends.

Alexey Pryakhin

Munich, October 2006.

Abstract

Modern automated methods for measurement, collection, and analysis of data in industry and science are providing more and more data with drastically increasing structure complexity. On the one hand, this growing complexity is justified by the need for a richer and more precise description of real-world objects, on the other hand it is justified by the rapid progress in measurement and analysis techniques that allow the user a versatile exploration of objects. In order to manage the huge volume of such complex data, advanced database systems are employed. In contrast to conventional database systems that support exact match queries, the user of these advanced database systems focuses on applying similarity search and data mining techniques.

Based on an analysis of typical advanced database systems — such as biometrical, biological, multimedia, moving, and CAD-object database systems — the following three challenging characteristics of complexity are detected: uncertainty (probabilistic feature vectors), multiple instances (a set of homogeneous feature vectors), and multiple representations (a set of heterogeneous feature vectors). Therefore, the goal of this thesis is to develop similarity search and data mining techniques that are capable of handling uncertain, multi-instance, and multi-represented objects.

The first part of this thesis deals with similarity search techniques. Object identification is a similarity search technique that is typically used for the recognition of objects from image, video, or audio data. Thus, we develop a novel probabilistic model for object identification. Based on it, two novel types of identification queries are defined. In order to process the novel query types efficiently, we introduce an index structure called Gauss-tree. In addition, we specify further probabilistic models and query types for uncertain multi-instance objects and uncertain spatial objects. Based on the index structure, we develop algorithms for an efficient processing of these query types. Practical benefits of using probabilistic feature vectors are demonstrated on a real-world application for video similarity search. Furthermore, a similarity search technique is presented that is based on aggregated multi-instance objects, and that is suitable for video similarity search. This technique takes multiple representations into account in order to achieve better effectiveness.

The second part of this thesis deals with two major data mining techniques: clustering and classification. Since privacy preservation is a very important demand of distributed advanced applications, we propose using uncertainty for data obfuscation in order to provide privacy preservation during clustering. Furthermore, a model-based and a density-based clustering method for multi-instance objects are developed. Afterwards, original extensions and enhancements of the density-based clustering algorithms DBSCAN and OPTICS for handling multi-represented objects are introduced. Since several advanced database systems like biological or multimedia database systems handle predefined, very large class systems, two novel classification techniques for large class sets that benefit from using multiple representations are defined. The first classification method is based on the idea of a k-nearest-neighbor classifier. It employs a novel density-based technique to reduce training instances and exploits the entropy impurity of the local neighborhood in order to weight a given representation. The second technique addresses hierarchically-organized class systems. It uses a novel hierarchical, supervised method for the reduction of large multi-instance objects, e.g. audio or video, and applies support vector machines for efficient hierarchical classification of multi-represented objects. User benefits of this technique are demonstrated by a prototype that performs a classification of large music collections.

The effectiveness and efficiency of all proposed techniques are discussed and verified by comparison with conventional approaches in versatile experimental evaluations on real-world datasets.

Zusammenfassung

Moderne Methoden zur automatischen Sammlung, Messung und Analyse von Daten in allen Bereichen der Industrie und Forschung liefern immer mehr Daten, deren Struktur darüber hinaus eine zunehmende Komplexität aufweist. Diese Komplexitätszunahme ist durch die folgenden zwei Aspekte begründet: erstens der Bedarf an präziseren Beschreibungen von Objekten der realen Welt, zweitens durch einen rapiden Fortschritt in Mess- und Analysetechniken, die eine vielseitigere Untersuchung von Objekten ermöglichen. Um sehr große Mengen solcher komplexen Objekte zu verwalten, werden hochentwickelte Datenbanksysteme eingesetzt. Im Gegensatz zu herkömmlichen Datenbanksystemen, die exakte Anfragen auf Objekten bearbeiten, konzentrieren sich die Benutzer von hochentwickelten Datenbanksystemen auf Ähnlichkeitssuche und Data Mining.

Ausgehend von einer Analyse der typischen hochentwickelten Datenbanksysteme, die biometrische, biologische, mobile, Multimedia- und CAD-Objekte verwalten, werden die folgenden drei grundlegenden Charakteristika festgestellt: Unsicherheit (probabilistische Merkmalsvektoren), multiple Instanzen (Mengen von homogenen Merkmalsvektoren) und multiple Repräsentationen (Mengen von heterogenen Merkmalsvektoren). Das Ziel dieser Doktorarbeit ist, Methoden für Ähnlichkeitssuche und Data Mining zu entwickeln, die mit unsicheren, multiinstantiierten und multirepräsentierten Objekten arbeiten können.

Der erste Teil der Arbeit beschäftigt sich mit Methoden der Ähnlichkeitssuche. Objektidentifizierung, wie z.B. Personenidentifizierung anhand von biometrischen Merkmalen, ist eine Methode der Ähnlichkeitssuche, die typischerweise zur Erkennung von Objekten in Bild-, Video- und Audiodaten eingesetzt wird. Wir entwickeln ein neues Wahrscheinlichkeitsmodell für Objektidentifizierung, das zwei neuartige Typen von Anfragen unterstützt. Zur effizienteren Bearbeitung dieser neuartigen Anfragetypen wird eine Indexstruktur eingeführt. Zusätzlich werden weitere Wahrscheinlichkeitsmodelle sowie Anfragetypen für probabilistische Multiinstanzobjekte und für probabilistische Beschreibungen von räumlichen Objekten spezifiziert. Unter Benutzung der Indexstruktur werden Algorithmen vorgestellt, die eine effiziente Bearbeitung dieser Anfragetypen erlauben. Die Praxisrelevanz von probabilistischen Objektbeschreibungen wird in einer realen Anwendung zur Ähnlichkeitssuche auf Videos demonstriert. Des Weiteren wird eine neue Technik vorgeschlagen, die aggregierte Multiinstanzobjekte verwendet und z.B. für die Ähnlichkeitssuche auf Videos geeignet ist. Dabei werden multiple Repräsentationen eines Objektes betrachtet, um die Effektivität der Suche zu erhöhen.

Der zweite Teil der Arbeit befasst sich mit zwei wichtigen Data Mining Techniken: Clustering und Klassifikation. Da die hochentwickelten Datenbanksysteme häufig in verteilten Umgebungen agieren, schlagen wir vor, durch den Einsatz von unsicheren Beschreibungen die Verschleierung der exakten Daten zu erreichen, um den Datenschutz beim Clustering zu gewährleisten. Um Multiinstanzobjekte zu clustern, wird ein modellbasiertes und ein dichtebasiertes Verfahren entwickelt. Des Weiteren behandelt die Arbeit zwei neue dichtebasierte Clusteringverfahren, die dichtebasierte Algorithmen um die Verarbeitung von multirepräsentierten Objekten erweitern. Zahlreiche hochentwickelte Datenbanksysteme im Molekularbiologie- oder Multimediabereich arbeiten mit bereits vordefinierten Klassensystemen oder Klassenhierarchien, die eine große Klassenanzahl aufweisen. Um mit solch hohen Klassenanzahlen und multiplen Repräsentationen effizient und effektiv umgehen zu können, entwickeln wir zwei neuartige Klassifikationsverfahren. Ausgehend von der Idee eines k-Nächsten-Nachbarn-Klassifikators wird eine neue dichtebasierte Technik zur Reduktion der Trainingsdaten erarbeitet. Außerdem wird eine Methode definiert, die Repräsentationen anhand der lokalen Entropie gewichtet und kombiniert. Das zweite, auf Klassenhierarchien agierende Klassifikationsverfahren verwendet eine neue Methode zur Reduktion großer Multiinstanzobjekte, wie z.B. bei Audiodaten, und setzt Support-Vektor-Maschinen für die Klassifikation ein. Die praktischen Vorteile dieser Methode werden an einem Prototyp für Klassifikation von großen Musiksammlungen demonstriert.

Effizienz und Effektivität aller vorgeschlagenen Verfahren werden ausführlich diskutiert und durch experimentelle Vergleiche mit herkömmlichen Methoden auf Daten aus realen Anwendungen verifiziert.

Contents

Part I

Preliminaries

Chapter 1

Introduction

Database systems have evolved from specialized computer applications to a central component of each modern computing environment. A *database system* is a system that describes, stores, and retrieves large amount of data. It consists of a *database management system* and a *database*. A database is defined as the collection of all stored data. A database management system is piece of software that controls accesses to a database. Moreover, it manages and updates a given database. Nowadays, database systems are employed for applications ranging from multimedia data management and location based services to computer aided design and science exploration. This broad use of database systems is a result of the natural evolution of information technology. The evolutionary path of the database functionalities can be divided into the following steps, illustrated in Figure 1.1: data collection and database creation, data management, advanced databases, and advanced data analysis (cf. [HK06]).

As illustrated in Figure 1.1, the database technology has evolved from primitive file processing to powerful database systems. Research and development efforts in databases from the 70s to the 80s have led from early hierarchical and network databases to the creation of widely accepted relational

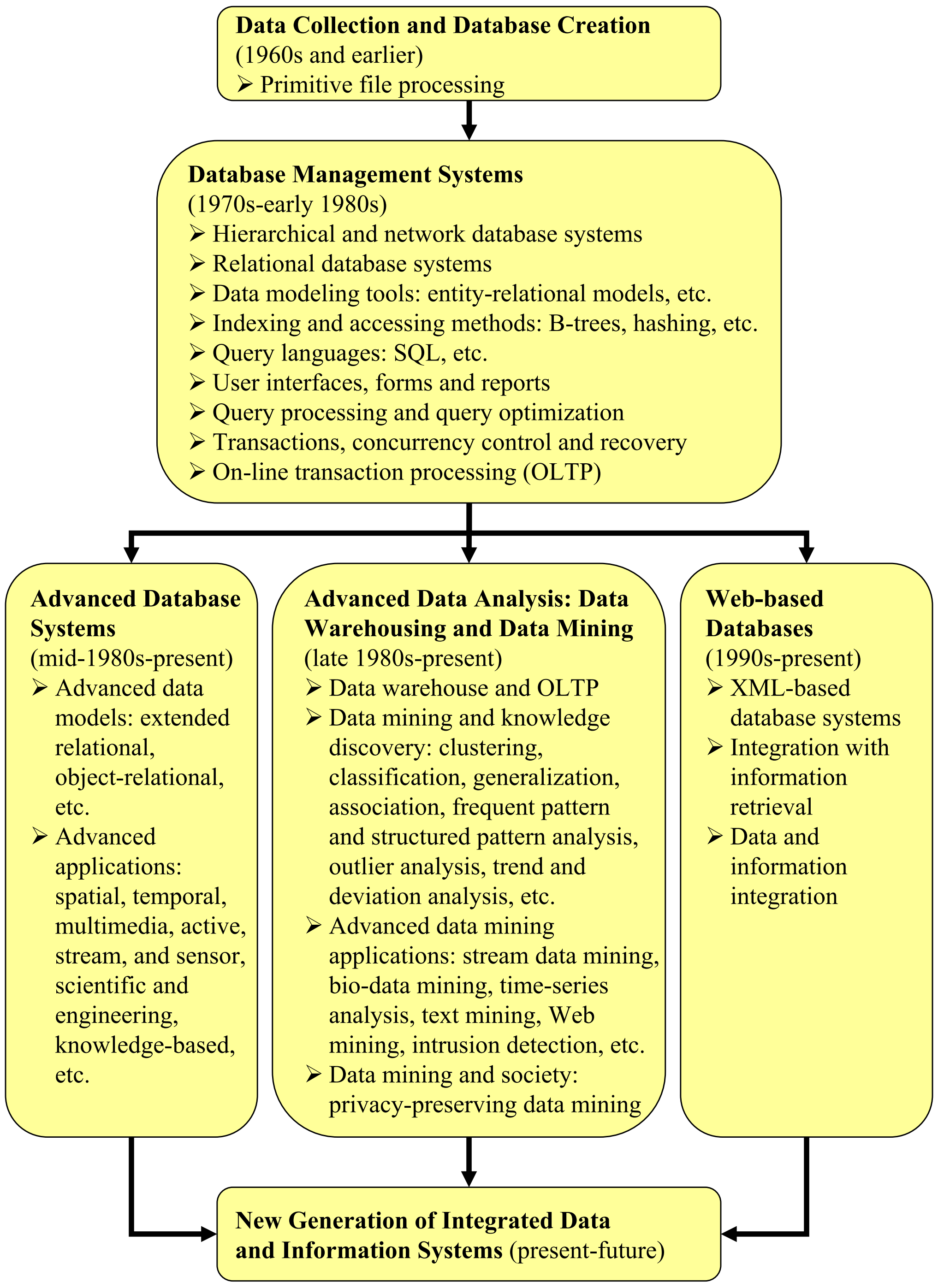

Data Collection and Database Creation
(1960s and earlier)
➢ Primitive file processing
Database Management Systems
(1970s-early 1980s)
➢ Hierarchical and network database systems
➢ Relational database systems
➢ Data modeling tools: entity-relational models, etc.
➢ Indexing and accessing methods: B-trees, hashing, etc.
➢ Query languages: SQL, etc.
➢ User interfaces, forms and reports
➢ Query processing and query optimization
➢ Transactions, concurrency control and recovery
➢ On-line transaction processing (OLTP)
Advanced Database Systems
(mid-1980s-present)
➢ Advanced data models: extended relational, object-relational, etc.
➢ Advanced applications: spatial, temporal, multimedia, active, stream, and sensor, scientific and engineering, knowledge-based, etc.
Advanced Data Analysis: Data Warehousing and Data Mining
(late 1980s-present)
➢ Data warehouse and OLTP
➢ Data mining and knowledge discovery: clustering, classification, generalization, association, frequent pattern and structured pattern analysis, outlier analysis, trend and deviation analysis, etc.
➢ Advanced data mining applications: stream data mining, bio-data mining, time-series analysis, text mining, Web mining, intrusion detection, etc.
➢ Data mining and society: privacy-preserving data mining
Web-based Databases
(1990s-present)
➢ XML-based database systems
➢ Integration with information retrieval
➢ Data and information integration
New Generation of Integrated Data and Information Systems (present-future)

Figure 1.1: The evolutionary phases of database technology (cf. [HK06]).

technology, efficient query processing, and transaction management. Since the mid-80s, database technology has created various data models such as extended relational, object-oriented, object-relational, and deductive models. Based on these sophisticated data models, a broad variety of application-oriented database systems — advanced database systems — have been developed. These advanced database systems include among others biometric, biological, spatial, temporal, multimedia, active, stream, sensor, scientific and engineering database and information systems for decision support and business intelligence. With the development of Internet-based global information systems, the role of distribution and data-sharing in database systems gets more and more important.

The advanced database systems require novel, application-oriented techniques for efficient storage, retrieval and management of large amounts of data. Furthermore, the rapidly growing, tremendous amount of data exceeds human ability to comprehend, overview and analyze the complex information stored in the advanced databases. As a result, *similarity search* and *data mining* techniques have become more and more popular in the past decade. In contrast to conventional exact-match queries usual for traditional database systems, similarity search finds data objects that differ only slightly from the given query object. "**Data Mining** refers to extracting or "mining" knowledge from large amounts of data"[HK06]. Alternatively, data mining can be seen as a step or a phase in the process of knowledge discovery as discussed below in Section 2.

The remainder of this chapter is organized as follows. First, we consider in Section 1.1 several advanced database systems and characteristics of data objects that are typical for these database systems from the application point of view. Section 1.2 and Section 1.3 introduce basics of similarity search and data mining techniques. Based on the demands of the advanced database systems, we elaborate in Section 1.4 the new challenges for similarity search and data mining techniques for advanced database systems.

1.1 Advanced Database Systems

This section aims at surveying database systems that have been established for advanced applications in the past decade. Furthermore, this section specifies several characteristics that are often required in these advanced database systems.

1.1.1 Biometric Database Systems

In recent years, biometrics has gained in importance enormously. Biometrics employs physiological or behavioral characteristics of a person in order to verify an identity. It distinguishes two major types of identification: physiological and behavioral. Physiological identification considers unique body characteristics such as the features of the iris, fingerprints, size and shape of a hand or a facial scan. Behavioral identification uses unique traits such as keystroke, a person's signature, and a voice scan. The major biometric technologies that are used nowadays are facial scan, finger scan, iris scan, hand scan, voice scan, retina scan, and signature scan (cf. [Deb04]).

Face recognition is one of the most widely applied technologies in person identification. Face recognition is broadly used in human-computer interfaces or in applications related to security, e.g. surveillance applications produce video sequences on which face identification is employed (cf. [Deb04]). Reasons for this broad usage are the availability of feasible technologies and the wide range of commercial and law enforcement applications.

Figure 1.2 illustrates a typical system architecture of a generic face recognition system using geometrical facial features (cf. [Deb05]). The input is a video stream that is captured by using a video camera. The output of the video capturing unit is later used by the face detection module where the position of a face is localized. In the next phase, facial features, e.g. width

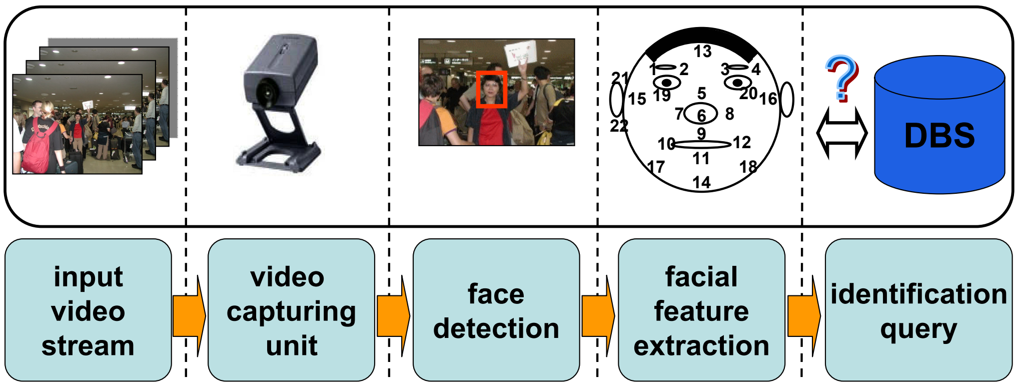


Figure 1.2: Face identification from video sequences.

of nose and depth of eye sockets, are extracted. The extraction of these features is based on feature points on the human face as depicted in Figure 1.2, e.g. left corner of the left eyebrow (point 1), left corner of the right eyebrow (point 3) and tip of the nose (point 6). Based on those facial feature values, an identification query can be performed according to the maintained database. Though several techniques for face recognition have been proposed recently, e.g. approaches using so-called "eigenfaces", local face features or line-based face recognition methods, it seems that techniques based on geometrical features provide the best accuracy of face recognition. This is shown in comparison to different approaches in [Deb04].

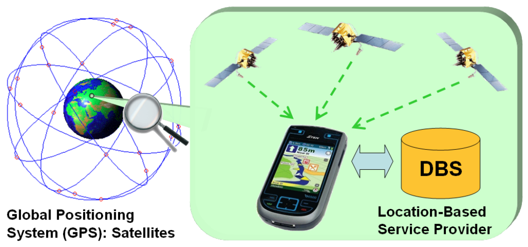
Although, the information extracted from facial images or even videos is very useful for person identification, the feature values are quite uncertain. This uncertainty can be individual for each feature and for each observed object. For instance, in a forensic image database some images may be taken under very controlled conditions while others are not, e.g. some images might be captured from a video of the surveillance camera at an airport. Depending on illumination aspect, pose, angle and distance to the camera or partial occlusion, some features may be measured with a varying degree of exactness. One further aspect influencing the exactness of measured feature values is the fact that a person changes appearance through aging, shaving a beard, consequences of a trauma, wearing glasses or simply through mood swings. Therefore, the use of biometrical features corresponds to dealing with the individual uncertainty of features and objects.

To sum up, the data objects stored in biometrical databases have two challenging characteristics. First, data objects are represented by feature vectors with a varying degree of exactness. Second, an individual can be described by different kind of features like face, fingerprint, and voice features, i.e. multiple representations of the same object are available.

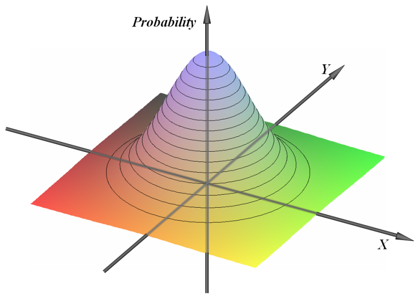
1.1.2 Moving Object Database Systems

In the last decade, rapid progress in wireless technology and miniaturization of computing devices has led to a broad usage of mobile computers instead of stationary desktops. Advances in mobile computer technology enable a wide range of novel applications like location-based services, mobile electronic commerce, tourist services and the digital battlefield. Several existing application types that will benefit from the development of mobile computer technology are transportation and traffic control, weather forecasting, emergency response and mobile resource management, cf. [MPV05] for more information. The key technique for all of the mentioned applications is the location management or the management of transient location information. The task of a moving object database system is the storage and management of location as well as other dynamic information about moving objects [MPV05].

In order to record the position of a moving object, the Global Positioning System (GPS) is typically used. More specially, GPS and telecommunication technologies sample the object position, i.e. the position is obtained at discrete time instances such as every two minutes. Figure 1.3(a) illustrates an example for GPS deployment in location based services. The accuracy of query results in moving object databases is conditioned by uncertainty


Global Positioning System (GPS): Satellites
DBS
Location-Based Service Provider

(a) Global Positioning System (GPS) and location-based services.



(b) Positional probability: the measurement error of a typical GPS module corresponds to a bivariate normal distribution.

Figure 1.3: Positional uncertainty in Global Positioning System (GPS).

caused by the measurement process in the sampling of positions. The error in a positional GPS measurement can be described by the bivariate Gaussian distribution, centered at the receiver's true antenna position as described in [PJ99]. The positional measurement error of a typical GPS module used in vehicle navigation systems is shown in Figure 1.3(b). More information about uncertainty in moving object databases can be found in [MPV05, PJ99, TWZC02].

To summarize, usually the accuracy of query results in moving object databases depends on spatial uncertainty in data. Conditioned by properties of GPS modules, this uncertainty can be described by a two-dimensional Gaussian distribution, centered at the true antenna position.

1.1.3 Sensor Network Database Systems

Recently, sensor networks are being used more and more frequently because of the rapid progress in sensor technologies. Sensors at multiple physical locations are employed in advanced applications such as environmental monitoring, weather prediction, process monitoring, medical monitoring, network

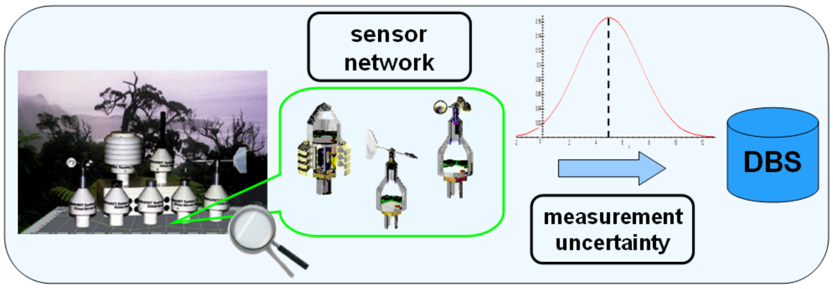


Figure 1.4: Uncertainty in sensor network databases caused by measurement error and transfer of a measured value after a certain time period has passed.

traffic flow controlling, flood prediction, and seismic detection. For instance, sensor networks in environmental monitoring collect data like temperature, noise level or CO_2 emissions, cf. Figure 1.4. Usually, the sensors send the collected information back to a database and due to bandwidth, power and storage limitations the exact values may only be obtained in certain time intervals. Again, if a query is posed at a time when the last exact value was recorded some time ago, the object value would become uncertain. The second factor causing uncertainty are measurement errors corresponding to physical properties of a given device. These measurement errors are usually modeled by a normal distribution, see e.g. [Zhu05].

1.1.4 Biological Database Systems

The huge volume and data-driven nature of modern experimental biology has led to a vast amount of databases that contain genomes, protein, and gene expression data. Because of the data driven character of modern biological research methods, researchers often retrieve or analyze information from these huge databases. The information is based on one or several aspects like annotation in textual form, amino acid sequence or 3D shape of a protein. Furthermore, bio-molecules are very complex objects according to

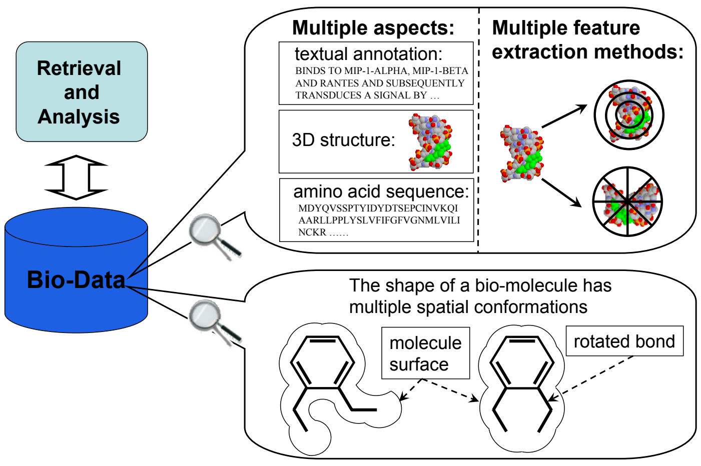

Retrieval and Analysis
Bio-Data
Multiple aspects:
textual annotation:
BINDS TO MIP-1-ALPHA, MIP-1-BETA AND RANTES AND SUBSEQUENTLY TRANSDUCES A SIGNAL BY …
3D structure:
amino acid sequence:
MDYQVSSPTYIDYDTSEPCINVKQI AARLLPPLYSLVFIFGFVGNMLVILI NCKR ……
Multiple feature extraction methods:
The shape of a bio-molecule has multiple spatial conformations
molecule surface
rotated bond

Figure 1.5: Retrieval and analysis in biological databases w.r.t. multiple characteristics and spatial conformations of bio-molecules.

their structure or their physical and chemical attributes. The more complex a data object is the more feature extraction methods exist that can be used to map the object to a representation suitable for data retrieval or analysis. As an example, the 3D shape of a bio-molecule can be described by volume, surface properties or features of 2D projections. Last but not least, bio-molecules are analyzed more precisely with respect to all of their possible spatial conformations [DLLP97a]. Bio-molecules may have different spatial conformations because they can adopt multiple shapes through rotation of their internal bonds as illustrated in Figure 1.5. Every angle combination of the rotatable bonds of a bio-molecule defines a spatial conformation.

We can summarize the challenging properties of biological databases as follows. First, there are different kinds of descriptions for the same biological object because of existing different aspects like a textual annotation or a sequence of amino acids, and using different feature transformation techniques like extraction of surface- or volume-oriented attributes for 3D structure. Second, a bio-molecule can be described by a set of feature vectors modeling different spatial conformations.

1.1.5 Database Systems for CAD-Data

In automotive and aerospace applications, a huge number of technical documents are generated during the development of complex engineering products. In order to support the development process, Computer Aided Design (CAD) is typically applied from the very first design to the final product. The demand for managing terabytes of data leads to the usage of database systems for CAD data management. To retrieve and analyze CAD-data, each object is usually mapped to a feature vector. There are several methods that extract feature values from 3-dimensional CAD-objects, e.g. [BKS+05a] gives an overview of these feature extraction methods. In CAD-catalogues, the parts are represented by some kind of 3D model like Bezier curves, voxels, polygon meshes, and additional textual information like descriptions of technical and economical key data. Furthermore, CAD-parts are usually decomposed into a set of spatial primitives — so-called covers — as described in [BKK+03]. The reasons for this decomposition are as follows: (1) Description of a CAD-object as a set of vectors is a generalization of the use of just one large feature vector. (2) The use of more sophisticated ways to model data can enhance both the effectiveness and the efficiency of applications using large amounts of data.

To summarize, we observe two frequent properties of CAD-data. First, an object is often given by multiple representations. Second, an object is modeled in a more natural and intuitive manner by taking a set of feature vectors into account.

1.1.6 Multimedia Database Systems

With the rapid development of digital technologies, computer networks and the Internet, the amount of multimedia data is growing enormously. One reason is that digital images, videos and pieces of music are easily copied and distributed. Thus, multimedia database systems (MMDBS) are employed in order to store such image, audio and video data. The typical fields for the usage of multimedia databases are content-based retrieval, video on demand systems, and speech or appearance based identification of individuals.

Multimedia objects like video clips or pieces of music can be considered as a sequence of scenes. If the order of the scenes is irrelevant, the multimedia object is represented by a set of scene descriptors. An example for such a setting is the clustering of news clips taken from different TV stations. Though, the order of the news might be varying, all stations will broadcast similar scenes of the current top stories.

Multimedia data such as movies can usually be viewed as objects with different kinds of descriptions, i.e. for each object there are multiple representations modeling different features of the object. For example, for music videos, we can collect audio features, such as pitch [TK00] or rhythm [TC02], and video features, such as color histograms or textures [AY99] as illustrated in Figure 1.6. Each of these multiple representations models a different aspect of a music video. Furthermore, the existence of different feature extraction techniques, like color and texture features, leads to the creation of multiple representations in multimedia object descriptions.

Unlike the traditional database systems which perform exact matches between the stored data and the query parameters, the MMDBS needs to handle the uncertain data and queries [CKG02]. The uncertainty is caused by the demand to consider high level or semantical features (cf. [CKG02, SN05] for details) in order to guarantee effectiveness in the retrieval of multimedia data, e.g. [Deb05] describes the usage of face detection and recognition techniques in order to calculate a compact description of video data. The uncertainty of describing and querying objects in MMDBS is also determined by the fact that the meaning of multimedia content is very often context sensitive. Furthermore, multimedia objects are usually audiovisual and, therefore,

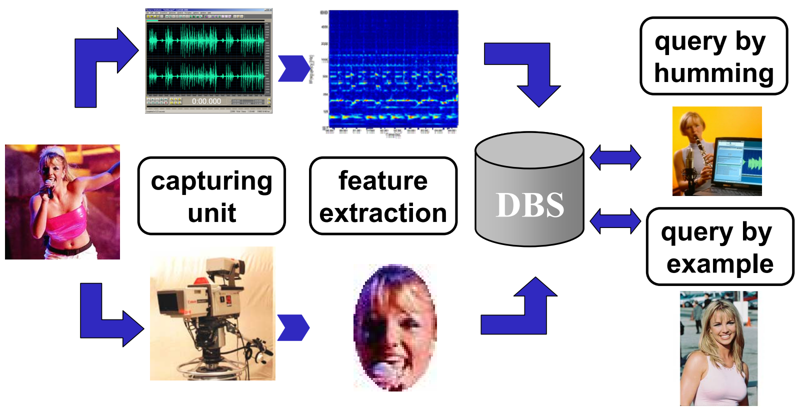


Figure 1.6: Feature extraction and query performing in multimedia database systems.

amenable to multiple interpretations conditioned by perception.

A typical MMDBS creates additional challenges due to the nature of multimedia data and the requirements of possible applications. We can summarize these challenges as follows. First, MMDBS must support the storage of large objects, because multimedia data such as audio or video can require gigabytes of storage. Thus, special preprocessing, storage and similarity search techniques are needed. Second, descriptions of multimedia objects are likely to be subjective and uncertain. Third, a multimedia object can be represented by a set of homogeneous descriptions — multiple instances. Fourth, the availability of versatile aspects and different feature transformations for multimedia data leads to the demand of handling multiple, heterogeneous descriptions — multiple representations — in retrieval and analysis methods.

1.1.7 Database Systems for Web Data

The invention of the World Wide Web (WWW) in the mid-1990s has created a demand for methods to gather, analyze, and utilize data available on the Web. Web data consist of HTML-documents that are published by an organization or an institution. Unlike common text documents, webpages provide a rich structure given by HTML tags. According to this structure, a webpage can be divided in a set of paragraphs. On the other hand, a set of webpages published by the same institution can be considered as a website as described in [EKS02]. Furthermore, a webpage or a website often contains different types of information like text, image or multimedia content.

In summary, we can observe two common characteristics of Web data. First, an object can be described by multiple aspects like text and images. Second, a webpage consists of several simple parts like paragraphs, and a website is given by a set of pages. Thus, web data are described in a more natural manner by using multiple instances.

1.1.8 Distributed Data

The progress in network technology, the globalization of industrial and scientific organizations, and the rapid increase of data volume and system scale have led to the fact that many data storage and processing systems are geographically dispersed. Therefore, not all data related to an application query or data analysis task is stored and managed in a centralized structure. Typical examples for geographically dispersed data sources include nationally or globally distributed branches of pharmaceutical research institutions operating on biological data, financial service institutions, or surveillance applications collecting and querying biometrical data.

Despite several advantages of distributed databases, e.g. including tremendous expansion in storage capacity, processing speed, and improvements in robustness, there typically exists a rapidly growing need for privacy preservation in distributed environments.

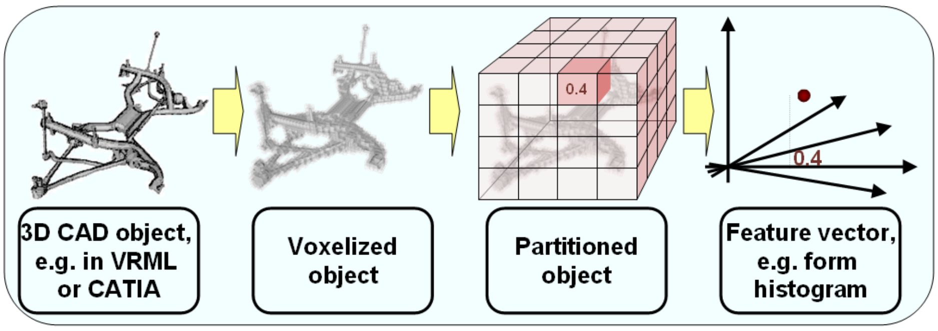

0.4
0.4
3D CAD object, e.g. in VRML or CATIA
Voxelized object
Partitioned object
Feature vector, e.g. form histogram

Figure 1.7: Feature extraction.

1.2 Similarity Search

Similarity search is an important technique in a broad range of applications like retrieval in multimedia, biological, biometrical, spatial, Web and CAD databases. To capture the similarity of complex domain-specific objects, the feature extraction is typically applied. The feature extraction aims at transforming characteristic object properties into feature values. Examples of such properties are the position and velocity of a spatial object, relationships between points on the face of a person such as the eyes, the nose, the mouth etc.The extracted values of features can be interpreted as a vector in a multidimensional vector space. This vector space is usually denoted as feature space F. The basic idea of a feature extraction on CAD data is demonstrated in Figure 1.7.

The most important characteristic of a feature space is that whenever two of the complex, application-specific objects are similar, the associated feature vectors have a small distance according to an appropriate distance function (e.g., the Euclidean distance). In other words, two similar, domain-specific objects should be transformed to two feature vectors that are close to each other w.r.t. the appropriate distance function. In contrast to similar objects, the feature vectors of dissimilar objects should be far away from each other. Thus, the similarity search is naturally translated into a neighborhood query in the feature space.

The two most important types of neighborhood queries in feature databases are:

Definition 1.1 (ε-Range Query) *:*
The user specifies a query object $q \in F$ and a query radius ε. The system retrieves all objects from the database that have a distance from q not exceeding ε. More formally: Let $\mathcal{DB} \subset F$ be a database consisting feature vectors from F and let $d : F \times F \longrightarrow \mathbb{R}$ be a similarity distance function. Then, the result RQ_ε of an ε-range query w.r.t. a query object $q \in F$ is defined as follows:

$$RQ_\varepsilon(q) = \{v \in \mathcal{DB} | d(q,v) \leqslant \varepsilon\}$$

Definition 1.2 (k-Nearest Neighbor Query (k-NN Query)) *:*
The user specifies a query object q and the cardinality k of the result set. The system retrieves those k objects from the database that have the least distance from q. More formally: Let $\mathcal{DB} \subset F$ be a database containing feature vectors from F and let $d : F \times F \longrightarrow \mathbb{R}$ be a similarity distance function. Then, the result NN_k of a k-nearest neighbor query w.r.t. a query object $q \in F$ is defined as follows:

$$\forall v \in NN_k(q), \forall w \in \mathcal{DB} \setminus NN_k(q) : d(q,v) < d(q,w)$$

For most types of feature spaces, there are multiple distance functions that are appropriate for similarity measure in certain applications. The most established type of distance functions are L_p distance metrics. L_p distance metrics in a d-dimensional feature space can be defined as follows:

Definition 1.3 (L_p Distance Metric)

$$L_p : \mathbb{R}^d \times \mathbb{R}^d \longrightarrow \mathbb{R}, L_p(x,y) = (\sum_{i=1}^{d} |x_i - y_i|^p)^{1/d}$$

where $p > 1$ and $x, y \in \mathbb{R}^d$

For $p = 2$ the L_p distance is called Euclidian distance function. The Euclidian distance function is very often used as similarity measure for feature spaces. The L_p distances are metrics because they fulfill the metric property that can be defined as follows:

Definition 1.4 (Metric Distance Function)
Let $\mu : \mathbb{R}^d \times \mathbb{R}^d \longrightarrow \mathbb{R}$ be a distance function. μ is called a metric iff:

1.
$$\mu(x, y) = 0 \Longleftrightarrow x = y \quad \forall x, y \in \mathbb{R}^d$$

2.
$$\mu(x, y) = \mu(y, x) \quad \forall x, y \in \mathbb{R}^d$$

3.
$$\forall x, y, z \in \mathbb{R}^d : \mu(x, z) \leqslant \mu(x, y) + \mu(y, z)$$

For an effective and efficient management of the feature vectors, several multidimensional index structures were proposed, e.g. R-Tree [Gut84] or R*-Tree [BKSS90]. Most feature databases are high-dimensional. Thus, the appearance of an effect called the “curse of dimensionality” causes particular performance problems for indexing structures. Therefore, a number of dedicated index structures for high-dimensional indexing have been proposed such as the X-tree [BKK96], the VA-file [WSB98], and the IQ-tree [BBJ$^+$00].

1.3 Data Mining

In the last decades, the amount of collected data in information and database systems has increased tremendously. To analyze this enormous amount of data, the interdisciplinary field of Knowledge Discovery in Databases (KDD) has emerged. The field of KDD combines disciplines like database systems,

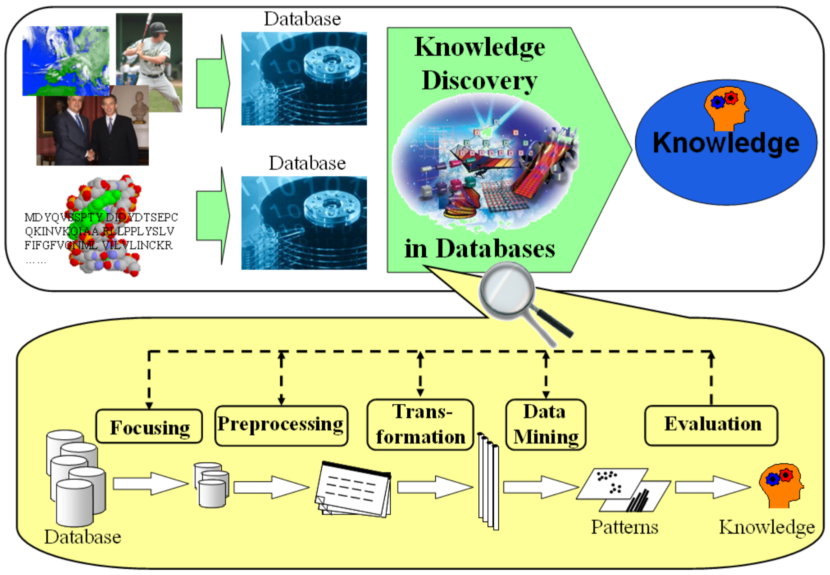


Figure 1.8: The KDD process — discovery of knowledge in large databases.

statistics, machine learning, visualization, and information science. In the following, the definition for the KDD process is proposed [FPSS96]:

Knowledge Discovery in Databases *is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.* Figure 1.8 gives a detailed overview of the KDD process and illustrates the basic flow of steps. Frequently, multiple iterations among these steps or even partial repeats of one or several steps are necessary.

1. **Focusing:** This step focuses on the definition of the goal for the particular KDD task. Another important aspect of this step is to determine the data to be analyzed and how to obtain and manage it.

2. **Preprocessing:** The main goals of this step are the integration, cleaning and (if necessary) completion of the data specified in the first step. The data from different sources should be integrated because they are typically obtained and maintained under considering different conven-

tions. A cleaning of the data is necessary in order to remove noisy or inconsistent data. Furthermore, the description of several objects may be incomplete. Thus, missing values or even missing attributes need to be completed.

3. **Transformation:** A further reduction of the, e.g. by selecting useful features or by using dimensionality reduction to minimize the effective number of attributes depending on the goal of the discovery task.

4. **Data Mining:** The goal of this essential step is to identify the relevant data mining task, e.g. clustering or classification. In this step, efficient and intelligent algorithms are used in order to extract novel, unknown and useful patterns from data.

5. **Evaluation:** The interesting patterns extracted in the previous step are prepared using knowledge visualization and representation techniques. In addition, the mined patterns are evaluated by domain experts according to the task definition.

The core step in the KDD-process is data mining. In [FPSS96], data mining is defined as follows.

Data Mining *is a step in the KDD process consisting of applying data analysis algorithms that, under acceptable computational efficiency limitations, produce a particular enumeration of patterns over the data.* The diverse data mining methods proposed recently in the literature can be categorized according to the following primary data mining tasks:

- **Clustering:** It groups objects of a database into classes (clusters) such that objects within one cluster are most similar to each other and objects of different clusters are most dissimilar to each other.

- **Outlier Detection:** The goal of these methods is to find data objects that do not correspond to the general behavior or model of the data.

- **Classification/Prediction:** Classification aims at assigning data objects to a subset of given classes. In order to perform the assignment, a function is typically learned on a small set of objects with known class assignments.

- **Association Rules:** The main task of these algorithms is to find association rules that show attribute-value conditions that occur frequently together in transaction databases. A transaction is a set of different items where each item has a different type.

- **Regression Analysis:** Regression methods estimate the relationship between the values of a target, or between the response variable and the predictor variable. Regression is related to the classification task because both learn a function from a training dataset.

- **Data Generalization:** The main goal of these algorithms is to derive a compact representation for a subset of data objects.

In the third part of this thesis, we focus on clustering and classification techniques. These very important data mining techniques are defined more precisely in the following.

Clustering *Clustering methods group objects of a database into clusters by maximizing the intra-cluster similarity and minimizing the inter-cluster similarity. Thus, after applying a clustering algorithm, similar objects are assigned to the same cluster and dissimilar objects to different clusters.*

Clustering aims at detecting new classes of data without any *a priori* knowledge. Thus, clustering is often also called *unsupervised learning* in contrast to classification where the classes are predefined and which is often also called *supervised learning.*

Classification *Classification algorithms learn a function that maps data objects to a subset of given classes. In order to learn this function, a small set of data objects with known assignment to subsets of classes C is provided. The process of learning is often denoted as training. The given set of assigned objects is usually called training set. Formally, a classifier is a function $\gamma : O \rightarrow C$ that maps each data object $o \in F \subseteq \mathbb{R}^d$ to its correct classes $c \subset C$.*

1.4 Advanced Database Systems: Challenges for Similarity Search and Data Mining Techniques

As demonstrated in Section 1.1, advanced database and information systems collect enormous amounts of data every day. In addition to the sheer amount of data, the complexity of data objects increases as well. Biological databases store more detailed information about molecules, multimedia applications store a huge amount of complex data consisting of images, audio and video, and HTML-documents provide embedded multimedia content which makes them much more complicated than ordinary text documents. The analysis of large collections of complex objects yields several new challenges to similarity search and data mining algorithms. In this thesis, we focus on the following challenges.

- **Uncertainty in the Object Description:**
  We argue that uncertainty is an important and challenging property appearing in advanced database systems. Let us consider the following advanced database systems which were introduced in more detail in Section 1.1. In applications of biometric or multimedia databases, the typical task is to identify individuals or objects according to fea-

tures which are not exactly known. Reasons for this inexactness are varying measuring techniques or environmental circumstances like illumination conditions. Since these circumstances are not necessarily the same when determining the features for different individuals, the exactness might strongly vary between the individuals as well as between the features. To identify individuals, similarity search on feature vectors is applicable, but even the use of adaptable distance measures is not enough to handle objects having an individual level of exactness. The uncertainty in object description appears also in many other advanced database systems like moving object, and sensor database systems, because no exact values to describe the data objects are available. Instead, the feature values are considered to be uncertain. This uncertainty is modeled by probability distributions instead of exact feature values. A typical application of such an uncertainty model are moving objects where the exact position of each object can be determined only at discrete time intervals. Queries often involve the position of objects between two time stamps or after the last known time stamp. Then, the objects are essentially uncertain unless the pattern of movement is very simple. The same problem exists, for instance, in sensor networks where continuously changing values such as temperature or wind speed can be measured at discrete time intervals only.

- **Multiple Representations:**
  Complex objects in several advanced database systems are often described by multiple representations modeling various aspects or generated by various feature extraction methods (cf. Section 1.1). We define a multi-represented object as an object that is described by a set of heterogeneous feature vectors. For example, for music videos, we can collect audio features such as pitch [TK00] or rhythm [TC02], and video features such as color histograms or textures [AY99]. Each of these multiple representations models a different aspect of a music

video. Obviously, the effectiveness of similarity search and data mining methods could greatly benefit from taking multiple representations into account. Further examples for multi-represented objects are proteins that can be described by text, amino acid sequences or 3D structures, webpages with multimedia content that can be described by feature vectors extracted from text and images, and CAD parts that are described by geometrical properties of 3D shape and 2D projections.

- **Multiple Instances:**
  In this thesis, we hold that the multi-instance character of data is a very frequent property of complex, domain-specific objects in advanced database systems. Recently, more and more advanced applications represent data objects as sets of feature vectors or multi-instance objects as discussed in Section 1.1. We define a multi-instance object as an object that is described by a set of homogeneous feature vectors. An example for a multi-instance object is a website which can be represented as a set of webpages. Other examples are CAD-parts represented by sets of voxels, molecules represented by sets of possible spatial conformations, and video clips considered as a set of images.

1.5 Outline of the Thesis

The major goal of this thesis is the development of novel techniques for similarity search, clustering and classification to cope with challenges of the advanced database systems as elaborated in Section 1.4 — uncertainty, multiple representations and multiple instances. The ideas and concepts presented in different chapters of this thesis are major extensions of material that has been published partially. For convenience, we list our own publications in Table 1.1. The main contributions of this thesis include:

Part II	Similarity Search Techniques	
Chapter 3	The Gauss-Tree: Efficient Object Identification in Databases of Probabilistic Feature Vectors.	[BPS06a]
Chapter 4	ProVeR: Probabilistic Video Retrieval using the Gauss-Tree.	[BGK$^+$07]
Chapter 5	Probabilistic Ranking Queries on Gaussians.	[BPS06b]
Chapter 6	Effective Similarity Search in Multimedia Databases using Multiple Representations.	[KKKP06]
Part III	Data Mining Techniques	
Chapter 7	Effective and Efficient Distributed Model-based Clustering.	[KKPS05]
Chapter 8	An EM-Approach for Clustering Multi-Instance Objects.	[KPS06]
Chapter 9	COSMIC: Conceptually Specified Multi-Instance Clusters.	[KPSZ06]
Chapter 10	Clustering Multi-Represented Objects with Noise. Hierarchical Density-Based Clustering for Multi-Represented Objects. Clustering Multi-Represented Objects Using Combination Trees.	[KKPS04a] [AKPS05] [AKPS06]
Chapter 11	Multi-Represented kNN-Classification for Large Class Sets.	[KPS05]
Chapter 12	Using Support Vector Machines for Classifying Large Sets of Multi-Represented Objects. MUSCLE: Music Classification Engine with User Feedback. Hierarchical Genre Classification for Large Music Collections	[KKPS04b] [BKK$^+$06] [BKKP06]

Table 1.1: List of own publications this thesis is based on.

- A novel probabilistic model for identification queries. This model is based on the assumption that the uncertainty of feature vectors can be modeled by Gaussian distributions.

- Novel types of queries called k-most-likely identification queries (k-MLIQ) and threshold identification queries (TIQ). These queries are based on the probability that a query object and a data object describe the same object.

- A general solution to calculate the probabilities that are necessary to process k-MLIQ and TIQ. These methods can be used in combination with several data structures and query algorithms.

- An index structure for efficiently processing k-MLIQ and TIQ called the Gauss-tree. The Gauss-tree belongs structurally to the R-tree family but uses novel algorithms for query processing, insertion and tree construction.

- The specification of two new types of probabilistic queries on sets of probabilistic feature vectors, and efficient algorithms for processing these new types of queries on sets of probabilistic feature vectors which are based on the Gauss-tree.

- Definition of a new model to handle uncertainty in spatial databases that does not rely on specifying guaranteed intervals, and introduction of a new useful type of probabilistic queries called probabilistic ranking queries (PRQs). In addition, algorithms for efficiently answering probability threshold queries and probabilistic ranking queries on the Gauss-tree are proposed.

- A novel similarity search approach for multi-represented multimedia objects using the fact that it is often beneficial to summarize multiple instances of a multimedia object for efficiency reason.

- A novel model-based distributed clustering algorithm that exploits uncertainty in order to achieve an arbitrary, predefined level of privacy-preserving.

- A model-based approach for statistical clustering of multi-instance objects that models instances as members of concepts in some underlying feature space.

- An unsupervised, density-based method for knowledge discovery in multi-instance datasets. This method specifies a multi-instance object by a set of so-called cluster attributes. In particular, we propose a method for hierarchical density-based clustering that avoids the creation of meaningless cluster attributes as well as a method for constructing a concept lattice based on concept attributes of different abstraction levels.

- Two density-based clustering algorithms extending DBSCAN and OPTICS for multi-represented data.

- A novel, density-based technique to reduce training instances for instance-based classification methods like k-NN classifier.

- An efficient algorithm for the classification of multi-represented data in an environment with large class sets.

- A novel *semi-supervised, hierarchical reduction of large multi-instance objects*. The demand for this technique is motivated by the fact that a multimedia object is usually described by a very large set of feature vectors, e.g. a song is represented by 10,000 to 50,000 feature vectors.

- An effective and efficient framework for *hierarchical genre classification* of music pieces in a *multi-representation* and *multi-instance* setting. Let us note that our framework can also be used for *genre classification* in flat class systems.

The remainder of this thesis is organized as follows.

In *Chapter 2*, we offer a general overview of existing similarity search techniques in the context of handling uncertain, multi-instance and multi-represented objects. We discuss several clustering and classification techniques that are able to handle multi-instance and multi-represented data. Furthermore, we introduce a few algorithms that are necessary for the understanding of the novel techniques developed in this thesis. In particular, we describe the notion of summarization techniques for multimedia objects, density-based clustering algorithms DBSCAN and OPTICS, and the model-based clustering algorithm EM. In addition, the chapter describes some basic notations used throughout this thesis.

Part II describes novel similarity search techniques, dealing with the three major challenges elaborated in Section 1.4.

In *Chapter 3*, we develop a comprehensive probabilistic theory in which uncertain observations of individuals or objects are modeled by probabilistic feature vectors (pfv), i.e. feature vectors where the conventional feature values are replaced by Gaussian probability distribution functions. Each feature value of each object is complemented by a variance value indicating its uncertainty. In addition, we define two types of identification queries, the k-most-likely identification and the threshold identification. For efficient query processing, a novel index structure, the Gauss-tree, is proposed. The experimental evaluation demonstrates that the probabilistic feature vectors (pfv) stored in a Gauss-tree significantly improve the result quality compared to traditional feature vectors. Additionally, it is shown that the Gauss-tree significantly speeds up query times when compared to other methods.

In *Chapter 4*, we consider content based video retrieval. An emerging and challenging topic in this area is the content based similarity search in video data. A video clip can be considered as a sequence of images or frames.

Since this representation is too complex to facilitate efficient video retrieval, a video clip is often summarized by a more concise feature representation. In this chapter, we transform a video clip into a set of probabilistic feature vectors (pfvs), i.e. a video clip is a multi-instance object where each instance is an uncertain description. In our case, a pfv corresponds to a Gaussian in the feature space of frames. We demonstrate that this representation is well suited for accurate video retrieval. The use of pfvs allows us to calculate confidence values for frames or sets of frames for being contained within a given video from the database. These confidence values can be employed to specify two types of queries. The first type of query retrieves the videos stored in the database which contain a given set of frames with a probability that is larger than a given threshold value. Furthermore, we introduce a probabilistic ranking query that retrieves the k database videos which contain the given query set with the highest probabilities. To efficiently process these queries, we introduce query algorithms on set-valued objects. The solution is based on the Gauss-tree. Our experimental evaluation demonstrates that sets of probabilistic feature vectors yield a compact and descriptive representation of video clips. Additionally, we show that our new query algorithms outperform competitive approaches on a database of over 900 real-world video clips.

In *Chapter* 5, we handle an uncertainty model that appears in spatial and sensor databases. A typical application of such an uncertainty model is moving objects where the exact position of each object can be determined only at discrete time intervals. Queries often involve the positions of objects between two time stamps or after the last known time stamp. One of the most important probability density functions for those applications is the Gaussian or normal distribution which can be defined by a mean value and a standard deviation. This chapter examines a new type of query on uncertain data objects, called probabilistic ranking queries (PRQ). A PRQ retrieves those k objects which have the highest probability of being located inside a given query area. To speed up probabilistic queries on large sets of uncertain data objects described by Gaussians, the index structure, the Gauss-tree(Chapter 3) is used. Furthermore, an algorithm for employing the Gauss-tree to answer PRQs is provided. The experimental evaluation demonstrates that the Gauss-tree achieves a considerable efficiency advantage w.r.t. PRQ compared to other applicable methods.

In *Chapter 6*, we deal also with similarity search in large multimedia databases. We propose a novel approach for similarity search in multimedia databases, taking multi-represented and multi-instance characteristics of multimedia objects into account. In particular, this chapter presents weighting functions to rate the significance of each representation for a given database object. This allows us to weight each representation during the query processing. A broad experimental evaluation shows the suitability and the effectiveness of multi-represented similarity search in music video databases.

Part III introduces several new clustering and classification approaches for handling the requirements of advanced database systems.

Chapter 7 focuses on the fact that many advanced application data are geographically dispersed, i.e. each site generates its own data and manages its own data repository. Analyzing and mining these distributed sources requires distributed data mining techniques to find global patterns representing the complete information. The transmission of the entire local dataset is often unacceptable because of privacy and security aspects, performance considerations, and bandwidth constraints. Traditional data mining algorithms that demand access to complete data are not appropriate for distributed applications. Thus, we suggest a distributed clustering algorithm that employs an uncertain data description for privacy preservation. In order to guarantee the privacy preservation, the proposed algorithm describes local models in terms of mixtures of Gaussian distributions. A broad experimental evaluation shows that the proposed framework is scalable in a highly distributed environment.

In *Chapter 8*, we present an expectation maximization approach to cluster multi-instance objects. A statistical process that models multi-instance objects is defined. Furthermore, we propose algorithms for estimating good initial segmentations of multi-instance datasets and for optimizing this segmentation by an expectation maximization algorithm. An experimental evaluation demonstrates that the new EM algorithm is capable of increasing the cluster quality for three real-world datasets when compared to results of existing clustering methods.

In *Chapter 9*, we propose a new method for clustering multi-instance objects, called COSMIC. COSMIC derives a hierarchy of so-called concept attributes corresponding to density-based clusters in the instance space. Afterwards formal concepts are derived, i.e. maximal sets of objects that can be described by a common set of concept attributes. Due to subset relations between sets, there is a natural relation between the concepts, connecting the set of all valid concepts into a so-called concept lattice. A broad experimental evaluation demonstrates that COSMIC outperforms other methods with respect to efficiency and cluster quality and is capable of extracting patterns from multi-instance datasets.

Chapter 10 presents an efficient density-based approach to cluster multi-represented data, taking all available representations into account. Therefore, we develop two different techniques to combine the information of all available representations dependent on the application. The evaluation part shows that the developed approach is superior to existing techniques. Afterwards, a hierarchical density-based clustering approach is introduced that distinguishes two basic types of representation semantics. To cluster multi-represented data we propose the use of combination trees for describing arbitrary semantic relationships between all representations. Furthermore, the hierarchical density-based clustering algorithm OPTICS is extended to the setting of multi-represented objects by employing combination trees. To support the usability of the proposed method, we present encouraging results for clustering two real-world datasets describing images and proteins.

In *Chapter 11*, we propose a new technique for the classification of multi-represented objects that is capable of distinguishing large numbers of classes. This technique can also cope with the data objects defined in general metric spaces. The proposed method is based on k-nearest-neighbor classification and employs density-based clustering as a new approach to reduce the amount of training data for instance-based classification. To predict the most likely class, the classifier employs a new method to use multiple object representations for making accurate class predictions. The introduced method is evaluated by classifying proteins according to the classes of Gene Ontology, one of the most established class systems for bio-molecules comprised of several thousand classes.

In *Chapter 12*, we present a novel content-based, hierarchical classification method to organize large music collections automatically, using a given genre taxonomy. To provide a versatile description of the music content, several kinds of features like rhythm, pitch or timbre characteristics are used, i.e. each piece of music is considered as a multi-represented object. Taking the highly dynamic nature of music into account, each of these features should be calculated up to several hundreds of times per second, i.e. each representation is described by multiple instances. Thus, a piece of music is represented by a complex object given by several large sets of feature vectors. The proposed approach is able to handle multiple characteristics of music content and efficiently achieves a high classification accuracy as shown in experiments performed on a real-world dataset.

Part IV concludes this thesis with a short summary.

Chapter 13 sums up and discusses the main contributions of the thesis. In addition, *Chapter 14* indicates some ideas for possible future work in the areas of similarity search and data mining for advanced database systems.

Chapter 2

Related Work

In this chapter, we discuss well established similarity search and data mining techniques that are used by our novel solutions. Furthermore, we survey topics that are related to this thesis. Section 2.1 begins with a description of conventional similarity search methods. In addition, we deal with similarity search methods that are applicable to uncertain, multi-represented, and multi-instance objects. In the last part of Section 2.1, we consider established summarization techniques for the aggregation of large multimedia objects like videos. Section 2.2 starts with a short explanation of basic clustering algorithms like EM, DBSCAN and OPTICS. Afterwards, we deal with established classification and clustering methods, that are capable of handling multi-instance and multi-represented objects.

2.1 Similarity Search

2.1.1 Similarity Search based on Feature Vectors

Similarity search for feature vectors is an important technique for information retrieval and data mining. Example applications include similarity search on structural features of 2D contours [MG93], 3D shape histograms for biomolecular objects [AKKS99], time series [FRM94, ALSS95], multimedia objects, and color histograms in image databases [NBE+93, FBF+94]. To compare different feature vectors, most systems employ a metric distance measure like the Euclidian distance. If some of the features are more important than others, the standard Euclidean query can be replaced by a weighted Euclidean query or a general ellipsoid query [NBE+93]. To increase the efficiency of similarity queries, various index structures have been proposed for high-dimensional feature spaces like [LJF94, BKK96]. For a survey cf. [BBK01].

In order to evaluate results of similarity search, we can use basic measures: precision and recall. Let A be a set of relevant objects. Let B be a set of retrieved objects. Then, $Precision = Card(A \bigcap B)/Card(B)$ and $Recall = Card(A \bigcap B)/Card(A)$.

2.1.2 Similarity Search based on Probabilistic Feature Vectors

Recently, the research on probabilistic queries over uncertain data has gained increasing attention. In [CKP03] a new uncertainty model is introduced and several new types of queries are described that allow the user to handle of uncertain data. This model is based on the assumption that it is possible to determine an interval for each feature value containing the exact value. Additionally, a feature value is described by an individual probability density function over this interval. We will refer to this model as the interval or spatial uncertainty model. [CXP+04] describes two methods for efficiently answering probabilistic threshold queries that are based on the R-Tree [Gut84]. A probabilistic threshold query (PTQ) returns all data objects that are placed in a given query interval with a probability exceeding a specified threshold value. The first of these methods does not rely on any assumptions about the underlying probability distributions. The second method is only suitable for a certain class of distribution functions, so-called symmetric and smooth variance monotonic density functions. The most prominent member of this class of distribution functions is the Gaussian distribution. The idea of this approach is based on precalculating so-called x-bounds. An x-bound limits an area in the value set for which it can be guaranteed that any interval being completely contained within this area has a probability of less or equal to $x\%$. For storing x-bounds the method exploits the observation that the behavior of two density functions of the same type only depends on a single parameter.

In [TCX+05], the U-Tree for indexing uncertain 2D objects was introduced. The paper relies again on the interval uncertainty model. In the U-tree, each object is guaranteed to be placed within a given polygon and a density function is given over this polygon. To index uncertain objects, the U-tree builds a conservative approximation for each node of an U-tree which consists of the minimum bounding rectangles (MBRs) of the polygons. The density functions are approximated by planes, starting at each side of a MBR. Besides the mentioned methods for indexing spatially uncertain objects, [DYM+05] introduces existential uncertainty. The idea of this approach is that the existence of each data object is uncertain. Thus, each object is coupled with the probability that it is indeed real.

2.1.3 Similarity Search based on Multiple Representations

Considering objects with multiple representations has attracted more and more attention in several research communities. A multi-represented object is described by a set of heterogeneous feature vectors. Each object $O \in \mathcal{DB}$ is represented by a given set of D representations $O_1, \dots, O_D$, where each representation is a feature vector in a separate feature space, i.e. $O_i \in F_i$

where $F_i \subseteq \mathbb{R}^{d_i}$, and $d_i \in \mathbb{N}$ denotes the dimensionality of the feature space of representation F_i $(1 \leq i \leq D)$.

In recent years, some work has been done on multi-represented similarity search in multimedia, biological and CAD databases. A common way to combine these representations is the use of a weighted linear combination. Various approaches have been proposed to compute the weights with the help of the user feedback. In order to give the feedback, an user has to label certain representations as relevant or irrelevant for the similarity. For instance, the approaches in [CLC98, RHM97] compute the weights based on the idea of relevance feedback. Authors of [AHCG00] suggest another relevance feedback based technique. This technique realizes a weighted distance approach that uses standard deviations of the features. From the user's point of view, it is rather inconvenient to provide feedback several times to get the result.

The interactive search fusion method [SJL+03a] provides a set of fusion functions, e.g. min, max, sum and product function that can be used for combining different representations in order to improve the effectiveness of similarity search. This method supports a manual and an interactive search that is supervised by the user's assistance or by a user-defined query. In addition, Boolean operators on aggregation functions are supported, e.g. "AND" can be applied to the product aggregation function. Let us note that this technique is supervised, i.e. it requires strong interaction with the user. This is not always desirable because for that the user has to understand the basic concepts of the method. In [NWH01a], a template matching method based on the time warping distance is presented. This approach can measure the temporal edit similarity. However, temporal order is not necessary in many applications. In addition, this technique is not applicable to large multimedia databases because it is linear in the number of feature vectors.

The authors of [BKS+04] introduce two methods for improving the effectiveness in a retrieval system that operates on multiple representation of

3D objects. The proposed techniques are based on the entropy impurity measure. The first method chooses the best representation w.r.t. a given query object. The second method performs so-called dynamical weighting of the available representations that is performed at query time, and that depends on entropy impurity in the local neighborhood of a query object. This work presents also encouraging experimental results that demonstrate for both proposed techniques a significantly improvement in effectiveness of the similarity search. The techniques, described in [BKS+04], need a set of labelled data in order to measure entropy impurity.

In [BKS05b], a pivot-based indexing schema for an efficient processing of similarity search queries on multi-represented objects and algorithms for efficient k-NN queries are presented. This method supports dynamically as well as statically parameterized distance functions. An example for a dynamically parameterized distance function is a weighted linear combination of distances in single representations while weights depend on a query object. In contrast to the dynamical parametrization, the statical parametrization relies on weights that are independent on query object, and are known in advance. The proposed solution is based on a combination of several pivot-based metric indices. The authors define the index structure, specify algorithms for performing k-NN queries on these index structures, and demonstrate a significant performance improvement in comparison to existing approaches.

2.1.4 Similarity Search based on Multiple Instances

A multi-instance object is an object that is described by a set of homogeneous feature vectors. More formally, each multi-instance object $O \in \mathcal{DB}$ is represented by a given set of D feature vectors $O_1, \ldots, O_D$, where each feature vector O_i is a vector in the same feature space, i.e. $O_i \in F$ where $F \subseteq \mathbb{R}^d$, and $d \in \mathbb{N}$ denotes the dimensionality of the feature space. Multi-instance objects are often called in literature as “set-valued” or “set of feature vectors”.

Thus, we use in this work denotations “multi-instance object”, “set-valued object”, and “set of feature vectors” as synonyms.

Multi-instance objects were usually handled by complex distance measures like [EM97, RB01]. In [EM97], the authors survey the following distance functions which are computable in polynomial time: the Hausdorff distance, the sum of minimal distances (SMD), the (fair-)surjection distance and the link distance. The Hausdorff distance is a metric, but does not seem to be suitable as a similarity measure, because it relies too much on the extreme positions of the elements of both sets. The last three distance measures are suitable for modelling similarity, but they are not metric. This circumstance makes them unattractive since there are only limited possibilities for processing similarity queries efficiently when using a non-metric distance function. In [EM97], the authors introduce a method for expanding the distance measures into metrics, but as a side effect the complexity of distance calculation becomes exponential. Furthermore, the possibility to match several elements in one set to only one element in the compared set is questionable when comparing general object sets with different cardinality.

Employing these distance measures for multi-instance objects, it is possible to pose similarity queries. However, the approach yields several problems. The selection of a suitable distance measure for a particular application is often quite difficult and the proposed distance measures for multi-instance objects often vary strongly when measuring the distance between multi-instance objects. Therefore, it is often necessary to try out a large variety of distance measures. Another problem is the understandability of the derived similarity distances. For complex distance measures and large multi-instance objects containing hundreds of instances, it is very difficult to understand why the multi-instance objects are similar. Finally, employing some of the distance measures yields efficiency problems. Since a considerable part of the distance measures for set-valued objects is non-metric, employing index structures is not always possible. Additionally, useful filter steps avoiding time consuming distance calculations (like a filter-based approach in [BKK+03]) were introduced for a minority of multi-instance distance measures only.

The *Hausdorff distance* is probably one of the best-known similarity measures for vector sets. For each element of both objects this method considers the nearest neighbor in the other object and takes the maximum of all these values as result. In other words, it identifies the most distant point of both sets and returns the distance to its closest point in the other set. The *Hausdorff distance* is a metric. More formally, let $d : F \times F \longrightarrow \mathbb{R}$ be a similarity distance function then the *Hausdorff distance* between two set-valued objects A and B can be defined as $H(A, B)$:

$$H(A, B) = \max(\max_{a \epsilon A} \min_{b \epsilon B} d(a, b), \max_{b \epsilon B} \min_{a \epsilon A} d(b, a)).$$

The *sum of minimal distances (SMD)* between two set-valued objects A and B is defined as $SMD(A, B)$ [EM97]:

$$SMD(A, B) = \tfrac{1}{2}(\tfrac{1}{|A|} \textstyle\sum_{i=1}^{|A|} (\min_{b \epsilon B} d(a_i, b)) + \tfrac{1}{|B|} \textstyle\sum_{i=1}^{|B|} (\min_{a \epsilon A} d(b_i, a))).$$

This function takes into account the distances between all elements and their nearest neighbor in the other vector set.

2.1.5 Summarization Techniques

Usually, multimedia objects like video clips or pieces of audio consist of thousands or even millions of feature vectors. For instance, a video consists of 25 or 30 frames per second. Thus, we get 1,500 or 1,800 feature vectors per minute of a video while describing each frame by a feature vector, e.g. color histogram. In order to handle such data efficiently, summarization techniques are usually applied to the original data, i.e. the original feature vectors are grouped together and each group is represented by a *summarization vector* or *summarization representative*. Then, similarity is defined by a suitable distance function based on these summarizations.

In general, we can distinguish two classes of summarization techniques, namely higher-order and first-order summarization. Higher-order summarization techniques are usually generated by applying optimization algorithms on feature vectors. They describe a video as a mix of statistical distributions or cluster representatives. In [GGM02], a higher-order summarization technique is presented which is based on Gaussian distributions or mixtures of Gaussian distributions. This technique describes spatial-temporal areas in a sequence of a few dozen frames by mixtures of Gaussian distributions.

The authors of [IBW$^+$04] demonstrate that Gaussian mixture models computed from video shots yield higher retrieval precision compared to keyframe-based models. The authors of [CSL99a] propose an approach for obtaining a compact representation of videos that computes the optimal representatives by minimizing the Hausdorff distance between the original video and its representation. If the Euclidian metric is used as distance function on the feature vectors, the k-means method [HK06] can be applied for summarization of video clip content [ZRHM98]. K-means minimizes the variance w.r.t. the representative vectors (this function is also called TD^2). In case of general metric spaces, the k-medoid method [HK06] can be applied for summarization. This method minimizes the distance between a video and its description.

First-order techniques calculate a small set of representative feature vectors as summarization vectors in order to describe a video. A randomized technique for summarizing videos, called video signature, is proposed in [CZ02a]. A video sequence in a database is described by selecting a number of its frames closest to a set of random vectors. This method requires only one single scan over the video or audio sequences and delivers a compact and reliable description that can be used for similarity search. The authors in [CZ02a] also propose a specialized distance function on the derived first-order summarization vectors.

2.2 Data Mining

2.2.1 Basic Clustering Approaches

In the past decades, many algorithmic solutions for the problem of clustering have been developed. In particular, the recent clustering approaches can be classified into the following categories (cf. [HK06]): *partitioning* methods like k-means or k-medoids, *hierarchical* methods like single-link clustering, *density-based* methods like DBSCAN, *grid-based* methods, and *model-based* methods like *Expectation Maximization (**EM**)*. In this section, we will provide a general overview of clustering approaches that are used in solutions proposed in this thesis.

Density-Based Methods. DBSCAN [EKSX96] is a density-based clustering algorithm where clusters are considered as dense areas that are separated by sparse areas. Based on two input parameters (ε and $MinPts$), DBSCAN defines dense regions by means of core objects. An object $o \in DB$ is called *core object* if its ε-neighborhood contains at least $MinPts$ objects. Usually, clusters contain several core objects located inside a cluster, and border objects located at the border of the cluster. In addition, objects within a cluster have to be "density-connected". DBSCAN is able to detect arbitrarily shaped clusters by a single pass over the data. To do so, DBSCAN uses the fact that a density-connected cluster can be detected by finding one of its core-objects o and computing all objects which are density-reachable from o. The correctness of DBSCAN can be formally proven (cf. lemmata 1 and 2 in [EKSX96], proofs in [SEKX98]).

OPTICS [ABKS99] extends the density-connected clustering notion of DBSCAN by hierarchical concepts. In contrast to DBSCAN, OPTICS does not assign cluster memberships but computes a cluster order in which the objects are processed. Additionally it generates the information which would be used by an extended DBSCAN algorithm to assign cluster memberships.

This information consists of only two values for each object, the core distance and the reachability distance. If the ε-neighborhood of an object o contains at least *MinPts* objects, the core distance of o is defined as the *MinPts*-nearest neighbor distance of o. Otherwise, the core distance is undefined. The reachability distance of an object p from o is an asymmetric distance measure that is defined as the maximum value of the core distance of o and the distance between p and o. Using these distances, OPTICS computes a "walk" through the dataset and assigns to each object o its core distance and the smallest reachability distance w.r.t. all objects considered before o in the "walk". In each step, OPTICS selects the object o having the minimum reachability distance to any already processed object. A special order of the database is generated according to its density-based clustering structure, the so-called cluster order which can be displayed in a reachability plot. A reachability plot consists of the reachability distances on the y-axis of all objects plotted according to the cluster order on the x-axis. The "valleys" in the plot represent the clusters since objects within a cluster have lower reachability distances than objects outside of a cluster.

Model-based Clustering. A model-based clustering algorithm assumes a model of clustering structures and calculates the best fit of the data to the given model. A very frequent applied model-based algorithm is EM [HK06]. Let us consider EM more closely.

Let $\mathcal{DB}$ be a set of d-dimensional points, i.e. $\mathcal{DB} \subseteq \mathbb{R}^d$. The general idea of the EM algorithm is to describe the data by a mixture model M of k Gaussian distributions, where k is the only input parameter. Instead of assigning each object to a cluster as it is the case for k-means-based clustering algorithms, EM assigns each object to a cluster according to a weight representing the probability of membership.

Each cluster $C \in M$ is a tuple $C = (\mu_C, \Sigma_C)$, where μ_C is the mean value of all points in C and Σ_C is the $d \times d$ covariance matrix of all points in C.

To compute the probability distributions, we need the following concepts.

The probability density of a point $\vec{x} \in \mathcal{DB}$ within a Gaussian density distribution $C = (\mu_C, \Sigma_C)$ is computed in the following way:

$$N_{\mu_C,\Sigma_C}(\vec{x}) = \frac{1}{\sqrt{(2\pi)^d|\Sigma_C|}} e^{-\frac{1}{2}(\vec{x}-\mu_C)^{\mathrm{T}}(\Sigma_C)^{-1}(\vec{x}-\mu_C)}.$$

The combined density for k clusters can then be computed by:

$$P(\vec{x}) = \sum_{i=1}^{k} w_{C_i} \cdot N_{\mu_{C_i},\Sigma_{C_i}}(\vec{x}),$$

where w_{C_i} is the fraction of points that belongs to cluster $C_i = (\mu_{C_i}, \Sigma_{C_i})$, i.e. w_{C_i} is the weight of C_i.

Then, the probability that a point $\vec{x} \in \mathcal{DB}$ belongs to a cluster C can be computed by the rule of Bayes:

$$P(C|\vec{x}) = w_C \frac{N_{\mu_C,\Sigma_C}(\vec{x})}{P(\vec{x})}.$$

The log-likelihood of a mixture model $M = (C_1, \ldots, C_k)$ of k Gaussian distributions which describes how good the model approximates the actual dataset can be computed by:

$$E(M) = \sum_{\vec{x} \in \mathcal{DB}} \log\left(P(\vec{x})\right).$$

The higher the value of $E(M)$, the more likely it is that the dataset $\mathcal{DB}$ corresponds to the mixture model M. Thus, the aim of the EM algorithm is to optimize the parameters of M in a way that $E(M)$ is maximized. For that purpose, the algorithm proceeds in four steps:

1. Initialization. Since the clusters, i.e. Gaussian distributions $C_1, \ldots, C_k$, are unknown at the beginning, a set of k initial clusters are generated randomly. For that purpose, each point $\vec{x} \in \mathcal{DB}$ is randomly assigned to a cluster C_i. An initial model is produced by computing μ_C and Σ_C for each cluster $C \in M$.

2. Expectation. Based on the current model, the parameters μ_C and Σ_C can be computed for each cluster $C \in M$ and the log-likelihood $E(M)$ of this mixture model M is obtained.

3. Maximization. In this step, $E(M)$ is improved through a recomputation of the parameters for each of the k clusters. Given a mixture model M, the parameters μ_C, Σ_C, and w_C of each cluster $C \in M$ are recomputed. The resulting mixture M' has an equal or higher log-likelihood than M, i.e. $E(M) \leq E(M')$. For improving the mixture, the parameters are recomputed as follows:

$$w_C = \frac{1}{|\mathcal{DB}|} \sum_{\vec{x} \in \mathcal{DB}} P(C|\vec{x}),$$

$$\mu_C = \frac{\sum_{\vec{x} \in \mathcal{DB}} \vec{x} \cdot P(C|\vec{x})}{\sum_{\vec{x} \in \mathcal{DB}} P(C|\vec{x})},$$

$$\Sigma_C = \frac{\sum_{\vec{x} \in \mathcal{DB}} P(C|\vec{x})(\vec{x} - \mu_C)(\vec{x} - \mu_C)^{\mathbf{T}}}{\sum_{\vec{x} \in \mathcal{DB}} P(C|\vec{x})}.$$

4. Iteration. Step 2 and 3 are iterated until the log-likelihood of the improved mixture model M' differs from the log-likelihood of the previous mixture M by a smaller value than a user specified threshold ε, i.e. until $|E(M) - E(M')| < \varepsilon$.

The result of the EM algorithm is a set of k d-dimensional Gaussian distributions, each represented by the mean value μ and the covariance matrix Σ and a weight w. The assignment of a point $\vec{x} \in \mathcal{DB}$ to a cluster C is given by the probability $P(C|\vec{x})$. Thus, we can compute how likely a point is assigned to each of the k clusters.

The log-likelihood of the result of the EM algorithm is usually dependent on the initial mixture model, i.e. on the model assumed in step 1, and on the number of clusters k. In [FRB98], a method for producing a good initial mixture is presented which is based on multiple sampling. It is empirically shown that using this method, the EM algorithm achieves accurate clustering results.

2.2.2 Classification and Clustering of Multi-Represented Objects

Classification of Multi-Represented Objects. In general, methods that employ multiple learners to solve a common classification problem are known as ensemble learning. An overview over ensemble learning techniques can be found in [VM02]. Within the area of ensemble learning, there is the subarea of classifier combination. The aim of the classifier combination is to use multiple independently trained classifiers and combine their results to increase the classification accuracy in comparison to the accuracy of a single classifier. Combining classifiers to learn from objects given by multiple representations has recently drawn some attention in the pattern recognition community [KHDM98, KBD01, Dui02]. The authors of [KHDM98] developed a common theoretical framework for combining classifiers which use multiple representations. Furthermore, the authors propose several combination strategies like max, min, sum, and product rule. [KBD01] describes so-called decision templates for combining multiple classifiers. The decision templates employ the similarity between classifier output matrices. In [Dui02], the author proposes a method of classifier fusion to combine the results from multiple classifiers for one and the same object. Furthermore, [Dui02] surveys the four basic combination methods and introduces a combined learner to derive combination rules offering better accuracy.

A related subarea of ensemble learning is the co-training or the co-learning which assumes a semi-supervised setting. The classification step of co-training employs multiple independent learners in order to annotate unlabeled data. [Yar95] and [BM98] were the first publications that reported an increase of classification accuracy by employing multiple representations. [Yar95] presents an unsupervised algorithm for sense disambiguation. The first representation is given by the local context of a word. The second representation contains the senses of other occurrences of that word in the same document. Both classifiers bootstrap each other iteratively. The authors of [BM98] train two Naive Bayes classifiers on independent representations of web pages. The first classifier uses the text representation of a web page. The second classifier works with the text of the hyperlinks referring to a page. New web pages are annotated by using one of these classifiers, and then are inserted into the set of labeled examples that is employed for the training of both classifiers. The authors report a significant increase in classification accuracy of both classifiers by iteratively retraining on the bloated training data. Recently, methods of hyper kernel learning [OS03] were introduced that are also capable to employ several representations for learning a classifier.

Clustering Multi-Represented Objects. The goal of clustering multi-represented objects is to find a global clustering for data objects that might have representations in multiple feature spaces. A similar setting to the clustering of multi-represented objects is the clustering of heterogenous or multi-typed objects [WZC+03, ZCM02] in web mining. In this setting, there are also multiple databases, each yielding objects in a separated data space. Each object within these data spaces may be related to an arbitrary amount of data objects within the other data spaces. The framework of reinforcement clustering employs an iterative process based on an arbitrary clustering algorithm. It clusters one dedicated data space while employing the other data spaces for additional information.

In [DS05], an algorithm for spectral clustering of multi-represented objects is proposed. The author proposes to calculate the clustering in a way that the disagreement between the cluster models in each representation is minimized. In [BS04], a version of Expectation Maximization (EM) clustering was introduced. Additionally, the authors proposed a multi-view version of agglomerative clustering. However, this second approach did not display any benefit against clustering single representations.

2.2.3 Classification and Clustering of Multi-Instance Objects

Classification of Multi-Instance Objects. Data mining in multi-instance or set-valued data objects has been predominantly examined in the classification section so far. In [DLLP97a] Dietterich et al. defined the problem of multi-instance learning for drug prediction and provided a specialized algorithm to solve this particular task by learning axis parallel rectangles. In the following years, new algorithms, increasing the performance for this special task, were introduced [Zho04]. In [WFP03], a more general method for handling multi-instance objects was introduced. It is applicable for a wider variety of multi-instance problems. This model considers several concepts for each class and requires certain cardinalities for the instances belonging to the concepts in order to specify a class of multi-instance objects. Additionally to this model, [GFKS02b] proposes more general kernel functions for comparing multi-instance objects.

Clustering of Multi-Instance Objects. For clustering multi-instance objects, it is possible to use distance functions for sets of objects like [EM97, RB01]. Having such a distance measure, it is possible to cluster multi-instance objects with k-medoid methods like PAM and CLARANS [NH94] or employ density-based clustering approaches like DBSCAN. Though this method yields the possibility to partition multi-instance objects into clusters, the clustering model consists of representative objects in the best case. Another problem of this approach is that the selection of a meaningful distance measure has an important impact of the resulting clustering. For example, netflow-distance [RB01] demands that all instances within two compared objects are somehow similar, whereas for the minimal Hausdorff [WZ00] distance the indication of similarity is only dependent on the closest pair.

2.2.4 Evaluation Techniques

Effectiveness measurement of a clustering method is a freguent task in this work. Thus, we describe here several approaches for this task. Often, we consider the agreement of the calculated clusterings to the given class systems. To do so, we can calculate different quality measures, e.g. precision, recall, F-measure and average entropy.

In order to calculate the precision and F-Measure, we proceed as follows. For each cluster c_i found by a clustering algorithm, its class assignment $Class(c_i)$ is determined by the class label of objects belonging to c_i that are in the majority. Then, we calculated the precision P, recall R or F-Measure within all clusters w.r.t. the determined class assignments by using the following formulas. $P = (\sum_{c_i \in C} \mathit{Card}(\{o | \mathit{Class}(o) = \mathit{Class}(c_i)\}))/Card(DB)$, $R = (\sum_{c_i \in C} Card(\{o | Class(o) \neq Class(c_i)\}))/Card(DB)$ and $\mathit{F\text{-}Measure} = (2 * \mathit{Precision} * \mathit{Recall})/(\mathit{Precision} + \mathit{Recall})$.

In addition, we can measure the average entropy over all clusters. This quality measure is based on the impurity of a cluster c_i w.r.t. the class labels of objects belonging to c_i. Let $p_{j,i}$ be the relative frequency of the class label $Class_j$ in the cluster c_i. We calculate average entropy as following.

$$Avg.Entropy = \sum_{c_i \in C} (Card(c_i) * (-\sum_{Class_j} p_{j,i} log(p_{j,i})))/Card(DB)$$

Furthermore, we can measure the agreement between the reference clustering and the results of a clustering algorithm using the Rand Index [HBV01], also known as Rand Statistics.

Part II

Similarity Search Techniques

Chapter 3

Efficient Object Identification

Object identification is a very important task in advanced database systems such as biometric and multimedia database systems. This chapter begins with the introduction of object identification in Section 3.1. Section 3.2 briefly discusses related work. In Section 3.3, we introduce the Gaussian uncertainty model for identification task. Based on this model, two novel query types are defined. The algorithms used to determine the exact results for both query types are described in Section 3.4. These algorithms can either be used on top of a sequential scan of the complete database or be used in the refinement step for the candidate set generated by our index structure, the Gauss-tree. Section 3.5 defines the Gauss-tree along with the methods for query processing and tree construction. In Section 3.6, we give a detailed experimental evaluation of both the effectiveness and the efficiency of our technique. Section 3.7 concludes this chapter.

3.1 Introduction

In many applications like face recognition [ZCPR03, CWS95], fingerprint analysis [oI84], or voice recognition [Cam97], data objects are represented by feature vectors with a varying degree of exactness or uncertainty (see Section 1.1 of Chapter 1 for details). Therefore, the observed feature values cannot be considered to be known exactly and two feature vectors describing the same object can be significantly different from each other. The degree of similarity between observed and exact values can vary from feature to feature because some features cannot be determined as exactly as others. For example, it is easier to determine the proportions of a face than the breadth of a nose. Additionally, to varying uncertainties between the features, we have to consider individual uncertainties for the objects as well because the circumstances in which a given data object is transformed into a feature vector may strongly vary. For example, most data collections consisting of facial images do not just contain images that were taken under the same illumination and having exactly the same distance between camera and face.

Due to these uncertainties, we are facing new problems. An object that is observed more than once under different circumstances will most likely generate a different feature vector for each of these observations. Thus, object identification, i.e. determining if two feature vectors belong to the same object, becomes much more complicated. For example, we might have a database of facial features. When observing one of the persons that are stored in this database, we cannot simple search for the observed feature vector in the database.

To solve identification problems, the simplest solution is to employ feature based similarity search. By defining a distance function like the Euclidian distance to feature vectors, we can assume that the distance between the feature vectors corresponds to the dissimilarity of objects. Thus, to identify an object, we could retrieve the nearest neighbor in the database. To speed up query processing for large databases, a variety of index structures for feature spaces of medium to high dimensionality has been proposed, e.g. the TV-tree [LJF94] and the X-tree [BKK96].

However, this solution does not consider the varying uncertainties between features and between objects. Thus, the nearest neighbor might be dominated by some very uncertain feature values and the retrieved object is not the correct one. To consider varying uncertainties among each feature, the Euclidean queries could be replaced by weighted Euclidean queries or general ellipsoid queries [SK97]. Though, these distance measures weight the importance of each features when comparing the objects, they assume the same level of uncertainty for all database object.

To handle the uncertainty of features and objects, we propose a new model to handle inexact data in databases. This model is based on the observation that the error of measurement for a feature value is assumed to follow a normal or Gaussian distribution for most applications. Thus, we call our model the Gaussian uncertainty model. The idea of this model is to extend a feature value $\mu_{i,j}$ for data object i by an uncertainty parameter $\sigma_{i,j}$ which is corresponding to the standard deviation describing the exactness of feature j. The complete probabilistic feature vector v_i is then associated to a multivariate Gaussian distribution N_{μ_i,σ_i}. Let us note that recently the concept of uncertainty was introduced in spatial temporal databases [CKP03, CXP+04]. However, the introduced concepts are not applicable to identification problems. We will discuss the differences in more details in Section 3.2.

3.2 Related Work

Recently, the research on probabilistic queries over uncertain data has gained increasing attention like [CKP03, CXP+04] (cf. survey in Section 2.1.2). **Why are recent spatial uncertainty models not appropriate for identification tasks?** The uncertainty model employed in [CKP03, CXP+04] allows to determine the probabilities that a given data object is placed in a given multi-dimensional interval within the query space. These probabilities are now used for a variety of queries, e.g. the already mentioned probability threshold queries. All of these queries are not directly applicable to identification tasks because the probability that two observations belong to the same data object cannot be determined by calculating the probability of containment within a certain multidimensional interval. Of course, we could assume that the query object is given by some multi-dimensional interval and retrieve the uncertain object in the database that provides the highest probability for being placed within this query interval. Besides the problem how to determine this interval for a given uncertain query, we now can apply the interval uncertainty model to object identification. However, the resulting method has several characteristics contradicting the intuition. Consider, for instance, a query object for which all features are known with a high degree of exactness: Therefore, this object has to be associated to a very small interval. Even if we find objects in the database which fit nicely to this query, the identification probability tends to be 0 with increasing exactness of the query. Inversely, if all features of the query object are known with little certainty, this would be modeled as a large interval by conventional uncertainty models, covering almost the complete data space. Therefore, all database objects have an identification probability of 100% in this model. To conclude, the probabilities of the interval uncertainty model are not applicable to intuitively modelling identification tasks.

We will show later that it is necessary to determine the identification probability using the Bayes' theorem in order to meet the intuition that identification probabilities should be close to 1 or close to 0 for exact knowledge of both query and database object (depending on how good the actual feature values fit) and be rather indifferent (tending to $1/n$ where n is the number of objects which *could* correspond to the query object) for knowledge which is less exact.

Similarity Search based on Feature Vectors. Similarity search for high dimensional feature vectors is an important technique for information retrieval and data mining. To compare different feature vectors most systems employ a metric distance measure like the Euclidian distance. If some of the features are more important than others the Euclidean query can be replaced by a weighted Euclidean query or a general ellipsoid query [NBE+93]. However, these approaches are not able to cope with individual uncertainty values for different objects.

3.3 The Gaussian Uncertainty Model for Identification Task

In this Section, we formally specify inexact object representations by the concept of probabilistic feature vectors (pfv). In addition, we define identification queries: threshold identification query (**TIQ**) and k-most-likely identification query (k-**MLIQ**). We finish this section with presentation of a two-dimensional example of probabilistic feature vectors.

3.3.1 Probabilistic Feature Vectors

A probabilistic feature vector v consists of d feature values μ_i and d uncertainty values σ_i where σ_i corresponds to the uncertainty of μ_i. The feature value μ_i is an observation e.g. from a sensor, and we assume that the measurement error of this sensor follows a normal distribution around the exact feature value with a known variance σ_i^2. Therefore, the data distribution of the observed values will follow a normal distribution N_{x_i,σ_i}, and the probability density that our feature value μ_i is observed, corresponds to $N_{x_i,\sigma_i}(\mu_i)$. Due to the symmetry of the Gaussians ($N_{x_i,\sigma_i}(\mu_i) \equiv N_{\mu_i,\sigma_i}(x_i)$), we can calculate $N_{\mu_i,\sigma_i}(x_i)$ to determine the probability density of the true feature value x_i for the observed feature value μ_i. This circumstance allows us to model an object by a multivariate normal distribution:

Definition 3.1 (Probabilistic Feature Vector (pvf))
A probabilistic feature vector v is a vector consisting of d pairs of feature values μ_i and standard deviations σ_i. Each pair defines a univariate Gaussian distribution of the true feature value x_i, defined by the following probability density function:

$$N_{\mu_i,\sigma_i}(x_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \cdot e^{\frac{-(x_i-\mu_i)^2}{2\sigma_i^2}}$$

The probability density of a probabilistic feature vector v for a given vector of actual values x can be calculated in the following way:

$$p(x|v) = \prod_{i=1}^{d} N_{\mu_i,\sigma_i}(x_i)$$

Our database DB consists of a set of n probabilistic feature vectors $v_i, 1 \leq i \leq d$ where d is the feature number.

3.3.2 Queries on a database of probabilistic feature vectors

Deriving a probability from a density function is usually done by integration over some interval. Thus, straightforward calculation of the probability that given a pfv, we will observe some query observation q always has a probability that tends to be 0 because we would integrate over an infinitely thin interval. However, for identification tasks we can employ the fact that a given observation has to belong to one pfv from a specified set. Thus, we now can use the theorem of Bayes. This theorem allows us to calculate the conditional probability that the query q belongs to a pfv v, under the condition that q belongs to one pfv of the set of all considered pfv in DB:

$$P(v|q) = \frac{P(v) \cdot p(q|v)}{\sum_{w \in \mathcal{DB}} (P(w) \cdot p(q|w))}$$

In this rule $p(x|v)$ is the probability density for observing x under the condition that we already observed v for the same data object. $P(v)$ ($P(w)$) is the general probability that v (w) is the answer to a query at all. In the following, we will assume that $P(v)$ ($P(w)$)is the same for any object and thus we can cancel it in the fraction. This assumption is based on the observation that it is usually not possible to anticipate the number of times that a certain object is queried.

Once we can determine this probability $P(v|q)$, we have a natural notion of how the queries for the Gaussian uncertainty model should be specified. The user can either specify a probabilistic query vector and a threshold for the probability. Then, the system has to retrieve all database objects which correspond to the query object with a probability of at least P_θ. We call this query a threshold identification query:

Definition 3.2 (Threshold Identification Query (TIQ))
Let q be a probabilistic feature vector and $P_\theta \in [0 \ldots 1]$ a probability threshold. The answer of a threshold identification query is defined as follows:

$$TIQ(q, P_\theta) = \{v \in \mathcal{DB} | P(v|q) \geq P_\theta\}$$

An example, for a TIQ is: Give me all persons in the database that could be shown on a given image with a probability of at least 10 %.

Similarly, we can also define a k-most-likely identification query, which retrieves the k database objects providing the highest probability of belonging to the database object:

Definition 3.3 (k-Most-Likely Identification Query (k-MLIQ))
Let DB be a database of probabilistic feature vectors v, let q be a probabilistic query vector and let $k \in \mathbb{N}$ be a natural number. Then, the answer to a k-most-likely identification query (k-MLIQ) on DB is defined as the

smallest set $MLIQ_k(x) \subseteq \mathcal{DB}$ with at least k elements fulfilling the following condition:

$$\forall v \in MLIQ_k(q), \forall w \in \mathcal{DB} \setminus MLIQ_k(q) : \\ P(v|q) > P(w|q)$$

An example k-MLIQ is: Give the the 10 most likely persons in the database that are shown on a given image.

We will show in Section and k-MLIQ can be answered in general. This general solution is either usable as a stand-alone solution operating on top of a sequential scan of the database DB. Additionally, our general solution can also be applied as a refinement step following after a filter step (e.g. by an appropriate index structure) for efficiency improvement. Several approximation techniques can be used as filter step, e.g. approximation by intervals. However, to guarantee correctness and completeness of the result, it is necessary to define a filter which guarantees no false dismissals (false hits are removed in the following refinement step). Therefore, an index structure guaranteeing no false dismissals is proposed in Section 3.5.

Figure 3.1 displays probabilistic feature vectors generated from 3 facial images of varying quality that are stored in a database and one for a query image. While feature F_1 is particularly sensitive to the rotational angle, F_2 is sensitive to illumination. The object O_1 is taken under good conditions (both features are relatively accurate), whereas for O_2 both rotation angle and illumination were bad. For O_3 the rotation was bad but the illumination was good. For the query object, in contrast, the rotation was good, but illumination was bad. We can easily recognize, that O_3 must be the object providing the highest probability for describing the same object as specified by the query. Our model derived in Section 3.4 will evaluate probabilities which correspond to this intuition: 77% for O_3 in contrast to 10% for O_1 and 13% for O_2. Therefore, a k-MLIQ with $k = 1$ would report O_3 as result. A TIQ with a threshold probability $P_\theta = 12\%$ would additionally report O_2.

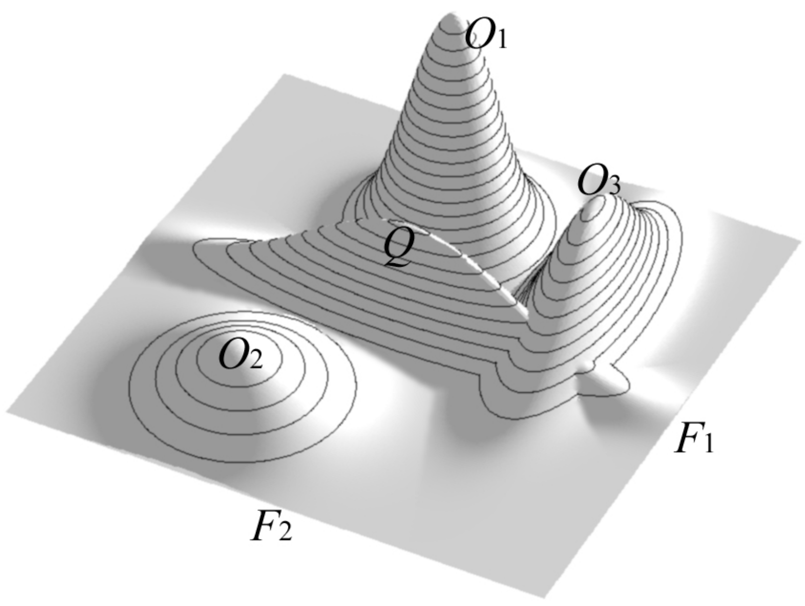


Figure 3.1: Probabilistic feature vectors in a 2D space. One query probabilistic feature vector and three database probabilistic feature vectors.

Since conventional similarity search does not consider the individual uncertainties, a similarity query using the Euclidean distance would obtain three rather similar distances ($d(Q, O_1) = 1.53$, $d(Q, O_2) = 1.97$, $d(Q, O_3) = 1.74$). Thus in our example, the nearest neighbor would be O_1 which is excluded when considering the variances. Thus, employing ordinary feature vectors cannot be used to draw conclusions about their probabilistic feature vectors having the feature vectors as mean vectors.

3.4 Processing of Identification Queries

To answer any query over a database DB of probabilistic feature vectors (pfv) with respect to a probabilistic query vector q, we have to model the probability that two probability distributions given by the query pfv q and a database pfv v correspond to the same true object. This yields again a probability density function $p(q|v)$. If the query object q would be an exact feature vector, we could calculate this probability density as mentioned in Section 3.3. However, if both objects are pfv, we have to consider all possible positions of the true feature vector when calculating $p(q|v)$. Then, the complete probability density corresponds to the integral over all these possible positions. Formally, we have to determine the probability density that a value x is the true feature value of both database and query object which implies the following term for each of the probabilistic features v_i and $q_i (i = 1, \ldots, d)$:

$$p(q_i|v_i) = \int_{-\infty}^{+\infty} p(v_i|x)p(q_i|x)dx$$

Remember that we are allowed to switch the mean value and the observed value due to the symmetry of the Gaussians. The term can be computed using the following lemma:

Lemma 3.1 (Joint Probability Density)
Let $v_i = (\mu_v, \sigma_v)$ be a probabilistic feature of a database object and $q_i = (\mu_q, \sigma_q)$ the corresponding probabilistic feature of the query object. Then, the joint probability density can be determined in the following way:

$$p(q_i|v_i) = \int_{-\infty}^{+\infty} N_{\mu_v,\sigma_v}(x) \cdot N_{\mu_q,\sigma_q}(x)dx = N_{\mu_v,\sqrt{\sigma_v^2+\sigma_q^2}}(\mu_q) \qquad (3.1)$$

Proof. *Based on the fact that $\sigma_v\sigma_q$ is a constant and using the Definition 3.1, the integral in the Equation 3.1 can be rewritten in the following way*

$$\int_{-\infty}^{+\infty} N_{\mu_v,\sigma_v}(x) \cdot N_{\mu_q,\sigma_q}(x)dx = \int_{-\infty}^{+\infty} \frac{\sigma_v\sigma_q}{2\pi\sigma_v^2\sigma_q^2}\mathbf{e}^{-(\mu_v-x)^2/(2\sigma_v^2)-(\mu_q-x)^2/(2\sigma_q^2)}dx$$

$$= \sigma_v\sigma_q \int_{-\infty}^{+\infty} \frac{1}{2\pi\sigma_v^2\sigma_q^2}\mathbf{e}^{-(\mu_v-x)^2/(2\sigma_v^2)-(\mu_q-x)^2/(2\sigma_q^2)}dx. \qquad (3.2)$$

Let us show that the following equation is satisfied.

$$\frac{1}{2\pi\sigma_v^2\sigma_q^2}\mathbf{e}^{-(\mu_v-x)^2/(2\sigma_v^2)-(\mu_q-x)^2/(2\sigma_q^2)} = \lambda \cdot \frac{1}{\sqrt{2\pi\sigma^2}}\mathbf{e}^{-(\mu-x)^2/(2\sigma^2)} \qquad (3.3)$$

If we apply the logarithm to the Equation 3.3, it follows that

$$\frac{(\mu_v - x)^2}{\sigma_v^2} + \frac{(\mu_q - x)^2}{\sigma_q^2} - 2\ln\frac{1}{2\pi\sigma_v^2\sigma_q^2} = \frac{(\mu - x)^2}{\sigma^2} - 2\ln\lambda \cdot \frac{1}{\sqrt{2\pi\sigma^2}}.$$

If we replace μ and σ^2 by

$$\mu = \frac{\mu_v \cdot \sigma_q^2 + \mu_q \cdot \sigma_v^2}{\sigma_v^2 + \sigma_q^2}, \qquad \sigma^2 = \frac{\sigma_v^2 \cdot \sigma_q^2}{\sigma_v^2 + \sigma_q^2} \tag{3.4}$$

and solve the resulting equation w.r.t. λ, it follows that the Equation 3.3 is satisfied iff

$$\lambda = \frac{1}{\sqrt{2\pi}(\sigma_v^2 + \sigma_q^2)}\mathbf{e}^{-(\mu_v - \mu_q)^2/(2(\sigma_v^2 + \sigma_q^2))}. \tag{3.5}$$

Using the Equation 3.3, the Equation 3.2 can be rewritten in the following way

$$\sigma_v\sigma_q\int_{-\infty}^{+\infty}\frac{1}{2\pi\sigma_v^2\sigma_q^2}\mathbf{e}^{-\cdots}dx = \sigma_v\sigma_q\int_{-\infty}^{+\infty}\frac{\lambda}{\sigma}\cdot\frac{1}{\sqrt{2\pi}\sigma}\mathbf{e}^{-(\mu - x)^2/(2\sigma^2)}dx.$$

Here, the first term in the integral is independent from the integration variable x. Therefore, the first term can safely be written before the integral (as it is a constant). The second term is the pdf of a normal distribution (with some complex values for μ and σ) which always integrates to 1 (when integrating from $-\infty$ to $+\infty$, independently of μ and σ). Therefore, we have

$$\sigma_v\sigma_q\int_{-\infty}^{+\infty}\frac{\lambda}{\sigma}\cdot\frac{1}{\sqrt{2\pi}\sigma}\mathbf{e}^{-\cdots}dx = \sigma_v\sigma_q\frac{\lambda}{\sigma}\int_{-\infty}^{+\infty}\frac{1}{\sqrt{2\pi}\sigma}\mathbf{e}^{-(\sigma - x)^2/(2\sigma^2)}dx = \sigma_v\sigma_q\frac{\lambda}{\sigma}\cdot 1.$$

If we replace σ and λ by their definitions in 3.4, 3.5 and exploit the fact that the resulting term corresponds to the normal distribution $N_{\mu_v,\sqrt{\sigma_v+\sigma_q}}(\mu_q)$, we get

$$\sigma_v\sigma_q\frac{\lambda}{\sigma}\cdot 1 = \sigma_v\sigma_q\sqrt{\frac{(\sigma_v^2 + \sigma_q^2)}{\sigma_v^2\sigma_q^2}}\frac{1}{\sqrt{2\pi}(\sigma_v^2 + \sigma_q^2)}\mathbf{e}^{-(\mu_v - \mu_q)^2/(2(\sigma_v^2 + \sigma_q^2))} =$$

$$\frac{1}{\sqrt{2\pi(\sigma_v^2 + \sigma_q^2)}}\mathbf{e}^{-(\mu_v - \mu_q)^2/(2(\sigma_v^2 + \sigma_q^2))} = N_{\mu_v,\sqrt{\sigma_v^2 + \sigma_q^2}}(\mu_q).$$

$\square$

The lemma allows us to calculate the probability that q and v correspond to the same data object by using μ_q as exact feature vector while increasing σ_v by σ_q. Thus, we have reduced the more general case to the easier case that one of the objects is exact and the other is a pfv. To calculate the $p(q|v)$, we have to combine all probabilities for all features and than again apply the rule of Bayes:

$$p(q|v) = \prod_{i=1}^{d} p(q_i|v_i)$$
$$P(v|q) = \frac{p(q|v)}{\sum_{w \in \mathcal{DB}} p(q|w)}$$

Employing this solution, we can give general algorithms for probability-threshold queries and k-maximum probability queries over a set S of probabilistic feature vectors. For the probability-threshold query, we first have to scan over S to determine the sum of the probability densities of all objects in S, i.e. the total probability. Afterwards, a second scan determines the actual probability $P(v|q)$ for each $v \in S$ and reports those with a probability above the threshold P_θ. For the k-MLIQ, a single scan over the database is sufficient, keeping those k objects (among all objects that have been processed so far) in a local list which have the highest probability density.

For the set S, we can use the whole database DB. In this case, we operate on top of a sequential scan of the database. As an alternative, we can also use a subset of DB which has been generated by a filter step, e.g. an appropriate index structure.

Properties. We conclude this section by briefly summarizing some properties of our solution in order to substantiate that the solution agrees with the intuitive requirements of the identification problem.

1. The sum of the probabilities of all retrieved objects of a TIQ or k-MLIQ cannot exceed 100%.

2. To obtain a high identification probability it is required that both database and query objects have a small uncertainty (σ_q, σ_v) and a high compliance of the observed features $(\mu_q \approx \mu_v)$, i.e. the Gaussians must have a high overlap and must be steep. Whenever we increase the uncertainty of database or query object (or both), the identification probability will decrease.

3. For very high uncertainty $(\sigma \rightarrow \infty)$ of the query or a database object (or both) our model becomes maximally indifferent, i.e. the identification probability corresponds to $1/n$ where n is the number of all possible objects.

4. If the Gaussian of a database object and that of the query object are quite disjoint, the identification probability is close to 0. Only in this case, it is possible that the identification probability slightly increases (up to $1/n$, see above) with increasing uncertainty because when increasing the uncertainty, the degree to which the object can be certainly *excluded from identification* decreases in this case.

3.5 The Gauss-Tree

In the previous Section, we have defined our basic notions of probabilistic feature vectors and queries on top of a set of such pfv. Derived from these basic definitions, we have introduced the basic algorithm for query processing on top of a sequential scan over an unordered file of pfv. The runtime complexity of these algorithms is linear in the number of stored objects. In the context of a large database, this is not acceptable, and we are now going to define the Gauss-tree, a suitable index structure improving k-most-likely identification and threshold identification queries on top of pfv.

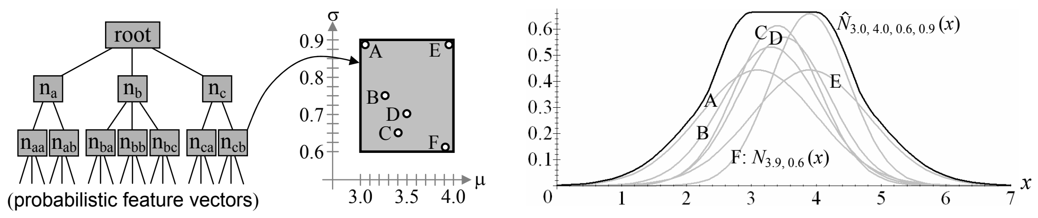


Figure 3.2: A 3 level Gauss-tree.

3.5.1 Structure of the Gauss-Tree

The Gauss-tree is a balanced tree from the R-tree family. In contrast to the other index structures from this family, not the space of the spatial objects (i.e. the Gaussians) is indexed but instead the parameter space $(\mu_i, \sigma_i, 1 \leq i \leq d)$ of the Gaussian. The structure of the index is inherited from the R-tree family which facilitates the integration into object-relational database management systems.

Definition 3.4 (Gauss-tree)
A Gauss-tree of degree M is a search tree where the following properties hold:

- *The root has between 1 and M entries unless it is a leaf. All other inner nodes have between $M/2$ and M entries each. A leaf node has between M and $2M$ entries.*
- *An inner node with k entries has k child nodes.*
- *Each entry of a leaf node is a probabilistic vector consisting of d probabilistic features (μ_i, σ_i).*
- *An entry of a non-leaf node is a minimum bounding rectangle of dimensionality $2d$ defining upper and lower bounds for every feature value $[\check{\mu}_i, \hat{\mu}_i]$ and every uncertainty value $[\check{\sigma}_i, \hat{\sigma}_i]$ as well as the address of the child node.*
- *All leaf nodes are at the same level.*

In Figure 3.2, we see an example of a Gauss-tree consisting of 3 levels. In the middle, we have depicted the minimum bounding rectangle of a leaf node for one of the probabilistic features. This minimum bounding rectangle allows to store feature values between $\check{\mu} = 3.0$ and $\hat{\mu} = 4.0$ and uncertainty values between $\check{\sigma} = 0.6$ and $\hat{\sigma} = 0.9$. A few sample pfv which are stored in this data page are also depicted. The Gaussian functions (probability density functions, pdfs) which correspond to these pfv are also shown on the right side of Figure 3.2 in gray lines.

For query processing, we need a conservative approximation of the probability density functions which are stored on a page or in a certain subtree. Intuitively, the conservative approximation is always the maximum among all (possible) pdf in a subtree. This maximum can be efficiently derived from the minimum bounding rectangle. In Figure 3.2, the maximum function which has been derived from the depicted minimum bounding rectangle is shown on the right side using a solid black line. As a formula, the approximating pdf $\hat{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(x)$ is given as:

$$\hat{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(x) = \max_{\mu \in [\check{\mu},\hat{\mu}], \sigma \in [\check{\sigma},\hat{\sigma}]} \{N_{\mu,\sigma}(x)\}$$

Since we assume independence in the uncertainty attributes, we can safely determine $\hat{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(x)$ in each dimension separately. Please note that $\hat{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(x)$ is not really a probability density function as it does not integrate to 1 for the whole data space. It is the conservative approximation of some probability density functions.

3.5.2 Query Processing

For efficient query processing, a closed formula for $\hat{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(x)$ without an explicit maximization process over two continuous variables is needed. This can be derived by the following lemma:

Lemma 3.2 *The conservative approximation $\hat{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(x)$ of the probability density functions stored in a data page can be exactly computed by the following piecewise function:*

$$\hat{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(x) = \begin{cases} N_{\check{\mu},\hat{\sigma}}(x) \text{ if } x < \check{\mu} - \hat{\sigma} & (I) \\ N_{\check{\mu},\check{\mu}-x}(x) \text{ if } \check{\mu} - \hat{\sigma} \leq x < \check{\mu} - \check{\sigma} & (II) \\ N_{\check{\mu},\check{\sigma}}(x) \text{ if } \check{\mu} - \check{\sigma} \leq x < \check{\mu} & (III) \\ N_{x,\check{\sigma}}(x) \text{ if } \check{\mu} \leq x < \hat{\mu} & (IV) \\ N_{\hat{\mu},\check{\sigma}}(x) \text{ if } \hat{\mu} \leq x < \hat{\mu} + \check{\sigma} & (V) \\ N_{\hat{\mu},x-\hat{\mu}}(x) \text{ if } \hat{\mu} + \check{\sigma} \leq x < \hat{\mu} + \hat{\sigma} & (VI) \\ N_{\hat{\mu},\hat{\sigma}}(x) \text{ if } \hat{\mu} + \hat{\sigma} \leq x & (VII) \end{cases}$$

Proof. *Since $\hat{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}$ is the maximum of some other Gaussian functions $N_{\mu,\sigma}(x)$ with mean values μ between $\check{\mu}$ and $\hat{\mu}$, the hull function is monotonically increasing for all $x \leq \check{\mu}$ and monotonically decreasing for all $x \geq \hat{\mu}$. Therefore, for a given x in the quadrants (I) to (III), the gaussian function which is maximal among all possible functions $N_{\mu,\sigma}(x), \mu \in [\check{\mu}, \hat{\mu}], \sigma \in [\check{\sigma}, \hat{\sigma}]$ must be on the left border of the minimum bounding rectangle, i.e. on the line parallel to the σ axis with $\mu = \check{\mu}$. We determine the σ value which maximizes $\hat{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}$ by setting the derivative with respect to σ to zero:*

$$\frac{\partial}{\partial\sigma} N_{\check{\mu},\sigma}(x) = 0$$

As the only positive solution we obtain a local maximum at:

$$\sigma_{max} = \check{\mu} - x$$

The function $N_{\check{\mu},\sigma}$ is also monotonically increasing with respect to σ for lower values of σ and monotonically decreasing for all $\sigma > \sigma_{max}$. For some x between $\check{\mu} - \hat{\sigma}$ and $\check{\mu} - \check{\sigma}$ our maximum is at the border of the minimum bounding rectangle, i.e. $\check{\sigma} \leq \sigma_{max} \leq \hat{\sigma}$, and therefore, the maximum value for some given x in quadrant (II) is

$$\hat{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(x) = N_{\check{\mu},\sigma_{max}=\check{\mu}-x}(x)$$

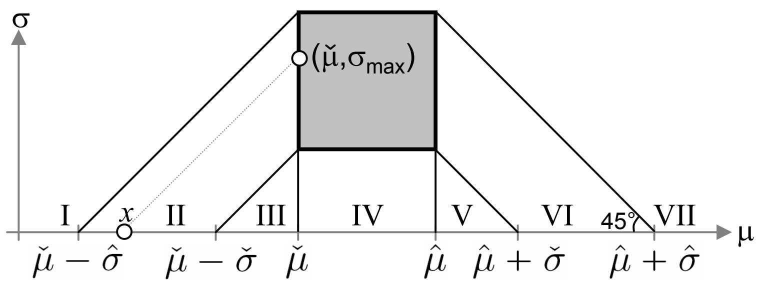

σ
(μ̌,σ_max)
I
x
II
III
IV
V
VI
45°
VII
μ
$\check{\mu}-\hat{\sigma}$
$\check{\mu}-\check{\sigma}$
$\check{\mu}$
$\hat{\mu}$
$\hat{\mu}+\check{\sigma}$
$\hat{\mu}+\hat{\sigma}$

Figure 3.3: The different sectors used to calculate $\hat{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(x)$.

In quadrant (I) the local maximum is at $\sigma_{max} > \hat{\sigma}$. Due to monotonicity, the global maximum (with restriction to the minimum bounding rectangle) must be at $\hat{\sigma}$. To the same reason, the maximum is at $(\check{\mu}, \check{\sigma})$ for all x in quadrant (III).

In quadrant (IV) the maximum $N_{\mu,\sigma}(x)$ is at $\mu = x$. For σ, we obtain to the same reason as for quadrant (III) a global maximum value of $\check{\sigma}$.

The cases (V) to (VII) are symmetric to (III), (II), and (I), respectively.

□

For query processing we will also need a lower bound $\check{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(x)$ for the stored Gaussian functions corresponding to the probabilistic feature vectors. This is defined by the following minimum:

$$\check{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(x) = \min_{\mu\in[\check{\mu},\hat{\mu}],\sigma\in[\check{\sigma},\hat{\sigma}]} \{N_{\mu,\sigma}(x)\}$$

It can be efficiently computed by considering only 4 Gaussian functions as stated in the following lemma:

Lemma 3.3 *The lower bound $\check{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(x)$ for all distance functions stored in* *a page given by the limits $(\check{\mu}, \hat{\mu}, \check{\sigma}, \hat{\sigma})$ can be computed by:*

$$\check{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(x) = \min\{N_{\check{\mu},\check{\sigma}}(x), N_{\check{\mu},\hat{\sigma}}(x), N_{\hat{\mu},\check{\sigma}}(x), N_{\hat{\mu},\hat{\sigma}}(x)\}$$

Proof. *When varying μ and σ in our function $N_{\mu,\sigma}(x)$ and fixing x, we observe only one local maximum and no local minimum (and no singularities). Therefore, the global minimum for the restricted function is at one of the four corner points of the rectangle delimited by $(\check{\mu}, \hat{\mu}, \check{\sigma}, \hat{\sigma})$. The four possible minima are tested.* □

Note that an even easier method is possible because it is very easy to decide whether the minimum is at $\check{\mu}$ or at $\hat{\mu}$ due to symmetry. All these methods have a constant time complexity.

Later, we will also need the approximation for the sum of all Gaussian functions which are stored in a data node or subtree. For this approximation, we consider the number of objects stored in the subtree n and apply:

$$n \cdot \check{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(x) \leq \sum_{t \in node} N_{\mu_t,\sigma_t}(x) \leq n \cdot \hat{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(x)$$

The accuracy of the approximation of the sum is bounded by:

$$n \cdot (\hat{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(x) - \check{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(x))$$

In our system, a query is defined by a probabilistic feature $q = (\mu_q, \sigma_q)$. The conservative approximations of the maximum, minimum, and sum can be determined analogously to Section 3.4 by the following equations:

- $\hat{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(q) = \hat{N}_{\check{\mu},\hat{\mu},\sqrt{\check{\sigma}^2+\sigma_q^2},\sqrt{\hat{\sigma}^2+\sigma_q^2}}(\mu_q)$
- $\check{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(q) = \check{N}_{\check{\mu},\hat{\mu},\sqrt{\check{\sigma}^2+\sigma_q^2},\sqrt{\hat{\sigma}^2+\sigma_q^2}}(\mu_q)$
- etc.

Note that although we have shown in this Section only the univariate case, it is very easy to extend all these formulas for the multivariate case because the individual univariate densities can be multiplied as we assume independence among the σ_i. This is also true for the lower and upper bounding pdf $\check{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(x)$ and $\hat{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(x)$ and for the sum approximation. Now we can provide the algorithms for our query types defined in Section 3.3 on top of the Gauss-tree.

3.5.3 k-Most-Likely Identification Query (k-MLIQ)

A most-likely identification query (MLIQ) reports the object for which the probability-based similarity is maximal. For the Gauss-tree, we give an algorithm operating on top of a priority queue [HS95]. The algorithm maintains a priority queue of pointers to some of the nodes (called *active nodes*) of the tree. The elements in the queue are ordered by the value of the approximation function evaluated for the query pfv. Let a be a node of the tree with $a.\text{appx} = (\check{\mu}_i, \hat{\mu}_i, \check{\sigma}_i, \hat{\sigma}_i, 1 \le i \le d)$ the μ and σ bounds associated to node a. Then the priority attribute a.prio of node a in the queue is defined as follows:

$$a.\text{prio}(q) = \hat{N}_{a.appx}(q) = \prod_{1 \le i \le d} \hat{N}_{\check{\mu}_i, \hat{\mu}_i, \sqrt{\check{\sigma}_i^2 + \sigma_{q,i}^2}, \sqrt{\hat{\sigma}_i^2 + \sigma_{q,i}^2}}(\mu_{q,i})$$

Intuitively, this ordering key $\hat{N}_{q.appx}$ corresponds to the maximum (relative) probability that one of the Gaussian functions stored in node a could yield when inserting the probabilistic query vector q. The top element of the queue is the node with maximum priority. Initially, the queue contains only the root. The algorithm runs in a loop which removes the top element from the queue, loads the corresponding node from disk (if not in cache), and reinserts pointers to the children (ordered by their priority attribute) into the queue. The algorithm keeps a *candidate* object in a variable which is the maximum pfv that has been seen so far by the algorithm in any of the leaf nodes. The algorithm stops when a probabilistic feature vector $v = (\mu_1, \sigma_1, \ldots)$ has

```
PriorityQueue kMLI_Query(int k, Page root, Point query) {
  // descending priority queues with k entries
  PriorityQueue candidates = new PriorityQueue(descending, k);
  // descending priority queues
  PriorityQueue activePages = new PriorityQueue(descending);
  //init
  activePages.put(root, MAX_REAL);
  //traverse Gauss-tree
  do {
      Page current = activePages.removeFirst();
      if(current is a data page) {
         for each vector in current {
             probability = calcProbability(vector, query);
             candidates.put(vector, probability);
           }
      } else {
        Page successors [] = current.getSuccessors();
        for each s from successors {
            probability = calcMaxProbability(s, query);
            activePages.put(s, probability);
        }
      }
  } while (activePages.isNotEmpty() &&
            candidates.getLastProbability ()
            < activePages.getFirstProbability());
  return candidates;
}
```

Figure 3.4: Pseudocode for the k-MLIQ.

been found for which the relative probability exceeds that of the top element t of the queue, with t.appx= $(\check{\mu}_i, \hat{\mu}_i, \check{\sigma}_i, \hat{\sigma}_i, 1 \le i \le d)$:

$$\prod_{1\le i\le d} N_{\mu_i,\sqrt{\sigma_i^2+\sigma_{q,i}^2}}(\mu_{q,i}) > \prod_{1\le i\le d} \hat{N}_{\check{\mu}_i,\hat{\mu}_i,\sqrt{\check{\sigma}_i^2+\sigma_{q,i}^2},\sqrt{\hat{\sigma}_i^2+\sigma_{q,i}^2}}(\mu_{q,i})$$

For k-MLI queries we have to maintain the set of k probabilistic feature vectors of maximal probability that have been found so far (the *candidate set*). The algorithm can safely stop now if all pfv in the candidate set have probabilities higher than the top element of the priority queue of the active nodes. The pseudocode is given in Figure 3.4.

3.5.4 Determining the Result Probability for k-MLIQ

The algorithm in Section 3.5.3 is able to determine those k elements having the highest probability with respect to the query object, but it is not able to determine the actual value of the probability. The reason is, that the stored Gaussian functions are only relative probabilities. These must be contrasted to the sum of the relative probabilities (theoretically) of *all* other Gaussian functions in the database, as discussed in Section 3.3:

$$P(t|q) = \frac{p(q|t)}{\sum_{s \in \mathcal{DB}} p(q|s)}$$

For pages which are far away from the query point, these relative probabilities (and also their approximations) are close to zero. Therefore, not *all* database objects need to be examined in order to determine the true denominator of this formula with sufficient accuracy.

We modify our algorithm of Section 3.5.3 in the following way:

- Whenever a leaf node is accessed, the corresponding pfv are examined and summed up for the total probability.
- Additionally, we maintain the upper and lower bounds for the impact of the objects which are stored in subtrees which have not yet been examined.
- The parent nodes of all subtrees which have not yet been examined are stored in the priority queue. Therefore, the upper and lower bounds are always updated whenever the top element is taken out of the queue and when the child nodes are re-inserted.
- The lower and upper bounds of the part of the sum which is caused by a single node in the tree in which n entries are stored, is given by $n \cdot \check{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(q)$ and $n \cdot \hat{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(q)$, respectively.
- The algorithm stops when both of the following conditions hold:

- The k pfv of highest probability are determined (i.e. all candidates have higher probabilities than the top element of the queue)
- The upper and lower bounds of the sum are close enough together to guarantee that the result is exact for all k answers according to user's specification of exactness (e.g. by a certain number of digits)

3.5.5 Threshold Identification Queries (TIQs)

This algorithm is similar to that of Section 3.5.4 with the difference that an unknown number of possible answer objects is maintained. An object must be stored in that set until it is guaranteed (according to the *lower* bound of the denominator) that the object has a probability which is below the specified threshold. The algorithm can safely stop when for all objects in the answer set it is guaranteed (according to the *upper* bound of the denominator) that they are safely above the specified threshold. We need both a lower and upper approximation of the Gaussian functions stored in a node.

If the user additionally specifies to report the actual probabilities of the answer elements at a specified accuracy, the algorithm may have to access more pages from the priority queue until all probabilities are known with sufficient certainty, like in Section 3.5.4

The pseudocode of our method for the probability threshold query is given in Figure 3.5.

3.5.6 Tree Construction

In the following we derive the optimization goals for the insert- and split strategies applied in the Gauss-tree. Intuitively, we have to collect such probabilistic feature vectors in one common leaf node (or subtree in general)

```
PriorityQueue TI_Query(Page root, Point query, float t) {
  real minSum, maxSum, sum = 0;
  PriorityQueue candidates = new PriorityQueue(desc);
  PriorityQueue activePages = new PriorityQueue(desc);
  //init
  activePages.put(root, MAX_REAL);
  minSum += root.minProb*root.size;
  maxSum += root.maxProb*root.size;
  //search
  do {
     Page current = activePages.removeFirst();
     minSum -= current.minProb*current.size;
     maxSum -= current.maxProb*current.size;
     if(current is a data page) {
        for each vector in current {
           probability = calculateProbability(vector, query);
           candidates.put(vector, probability);
           sum += probability;
        }
     } else {
       Page  successors [] = current.getSuccessors();
       for each s from successors {
           probability = calculateMaximalProbability(s, query);
           activePages.put(s, probability);
           minSum += s.minProb*s.size;
           maxSum += s.maxProb*s.size;
       }
     }
     //delete unnecessary candidates
     while(candidates.getLastProbability()/(minSum+sum) < p)
      candidates.removeLast();
  } while (activePages.isNotEmpty()
     && activePages.getFirstProbability()/(minSum+sum)<p);
  //calculate final probabilities
  for each c in candidates {
   prev = candidates.getProbability(c);
   candidates.updateProbability(c,  prev/(maxSum+sum));
  }
  return candidates;
}
```

Figure 3.5: Pseudocode for the TIQ.

which share both similar μ and σ values because if one of these pfv is needed for a given query, also the other ones are probably needed for that query. However, the situation is not that clear as it is for conventional feature vectors where the typical optimization goal is to achieve hyper-rectangles with approximately uniform side lengths. The main difference is the following: If we have a node which contains only pfv which have a small standard deviation for one of the probabilistic features, i.e. $\hat{\sigma}_i \simeq 0$ then it is also beneficial if the μ values are spread over a small range, i.e. $\hat{\mu}_i - \check{\mu}_i \simeq 0$ because if we have both small values of σ as well as small *ranges* of μ then this node will be very selective, i.e. the node will only be accessed for queries for which the stored pfv are highly probable candidates. In this case, $\hat{N}_{\ldots}(x)$ is narrow, and unnecessary page accesses can be avoided. In contrast, if the node also contains pfv with a high variance then a small range of μ will not help much either because $\hat{N}_{\ldots}(x)$ will be spread over a wide range anyway. But if the range of σ values (i.e. $\hat{\sigma}_i - \check{\sigma}_i$) is small, then we know at least that this node contains no pfv with a high probability density. In this case, the node can be excluded for many queries (e.g. k-MLIQ) which have already found at least k pfv with higher probability in some other nodes of the Gauss-tree.

We can summarize this intuition for the split strategy (on every node overflow) in the following way: If $\hat{\sigma}_i$ is low, then perform a node split according to μ_i. Otherwise perform a split operation according to σ_i. In the following, we will capture this intuition more precisely because we do not only have to decide whether to split in μ or σ but also which of the d different μ or σ have to be used for splitting. Additionally, our analysis gives a formal justification for the strategy. The mathematical model can be used not only for the decision of the split but also for resolving the situations during the insert (i.e. whenever more than one branch of the tree is eligible for the new pfv).

The split decision must minimize the probability of a node to be accessed for an arbitrary query. This probability is proportional to the integral of the hull curve:

$$\int_{-\infty}^{+\infty} \hat{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(x)dx$$

The integral can be determined for each probabilistic feature separately. The computation of the integral is straightforward. Remember the case analysis of lemma 3.2. Case (IV) is a constant function, and cases (I), (III), (V), and (VII) are Gaussian functions with given μ and σ for which efficient integration methods are known. We apply sigmoid approximation by a degree-5 polynomial in order to calculate the integral efficiently. The only part which requires a little bit of consideration is case (II) and its symmetric counterpart (VI) where we have to integrate over $N_{\check{\mu},\check{\mu}-x}(x)$ from $\check{\mu}-\hat{\sigma}$ to $\check{\mu}-\check{\sigma}$. However, substituting $(\check{\mu}-x)$ for σ in the definition of the probability density function of the Gaussian distribution yields:

$$N_{\check{\mu},\check{\mu}-x}(x) = \frac{1}{\sqrt{2\pi e}\cdot(\check{\mu}-x)}$$

which integrates to $(\ln\hat{\sigma} - \ln\check{\sigma})/\sqrt{2\pi e}$ for the above mentioned integration limits.

For the insertion strategy, we apply the following rules to select a path of the Gauss-tree :

- If the new pfv fits into exactly one node, this node is followed.
- If the new vector does not fit into any node, we examine all subnodes and find the leaf node which causes the least increase of volume.
- If the new vector fits into more than one node, we follow all paths and try to find a leaf node where the node exactly fits in (or minimize the increase of volume, if no exactly fitting node exists).

When a node is beyond its capacity, it has to be split. We tentatively perform a median split in each μ-dimension and each σ-dimension of the Gauss-tree. For every tentative split, we determine the lower and upper μ and

σ bounds of the two resulting nodes, and evaluate the integral $\int \hat{N}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(x)dx$ for both nodes. The split operation minimizing the sum of these two integrals is made permanent.

3.6 Experimental Evaluation

For our experimental evaluation, we implemented the Gauss-tree and all compared methods in Java. All experiments described below were performed on a workstation that is equipped with two AMD Opteron 1.8 GHz processors and 8 GB main memory. We used up to 50 MByte as database cache which was cold started before each experiment. In our experiments, we compare the effectiveness and efficiency of the proposed solution for handling uncertain data on two different datasets.

Dataset 1 consists of 10,987 27-dimensional color histograms of an image database. To describe these images as probabilistic feature vectors, we complemented each dimension with a randomly generated standard deviation. A total number of 100 objects was randomly selected and new observed mean value was generated w.r.t. the corresponding Gaussian. For these queries, new standard deviations were randomly generated. To additionally test our method on a larger dataset, we randomly generated 100,000 probabilistic feature vectors in a 10-dimensional feature space along with corresponding σ values (dataset 2). For dataset 2, 500 query vectors were selected and modified as described above.

To demonstrate that ordinary similarity search on feature vectors using Euclidean distance is not sufficient for probabilistic queries on imprecise data, our first experiment compares precision and recall for both methods. To compare the performance of the Gaussian uncertainty model to ordinary similarity search, we processed an MLIQ on the Gauss-tree and a nearest neighbor query on the mean values. For each query, we measured precision

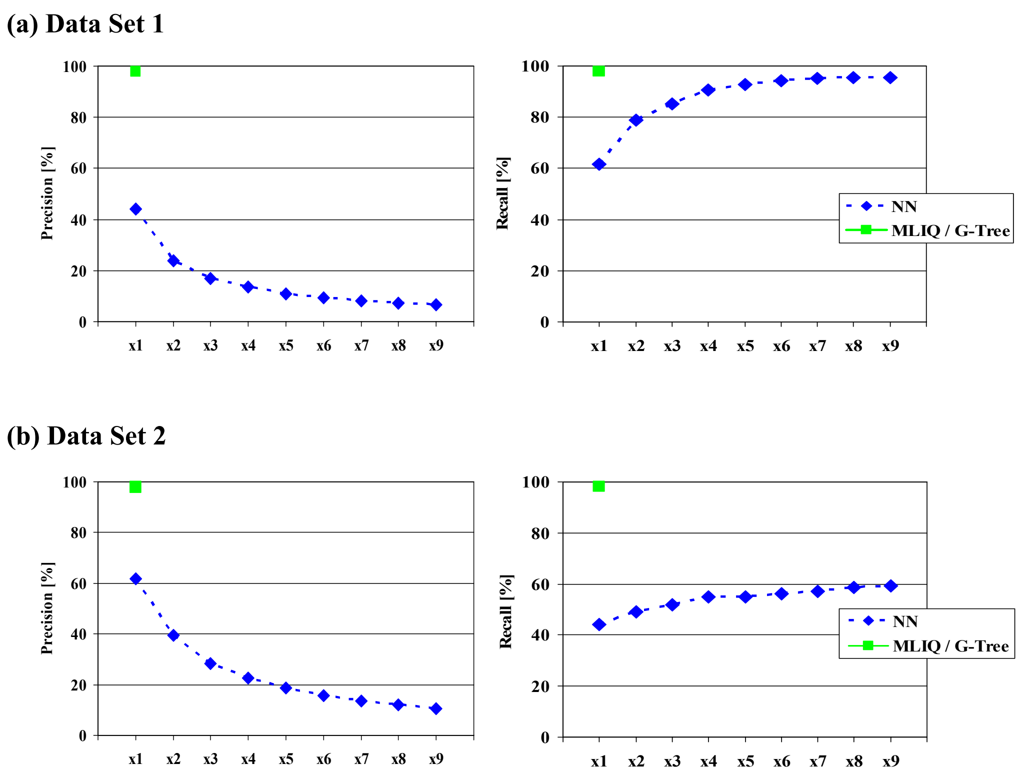


Figure 3.6: Precision and Recall of 3-NN query on conventional feature vectors and 3-MLIQ on pfv.

and recall. The recall is the percentage of retrieved and correct answers among all objects in the database that are correct and precision is the percentage of the retrieved and correct objects among all objects in the result set. For NN queries and MLIQ, both measures are the percentage of queries that retrieved the correct object.

Figure 3.6 compares precision and recall for both methods. The MLIQ achieved a precision and recall of 98% for dataset 1 and 99% for dataset 2. Thus, our new query, based on the Gaussian uncertainty model, achieved almost optimal results in this experiment. On the other hand, the NN query displayed only a precision of 42% for dataset 1 and 61% for dataset two. Thus, ordinary similarity search seems not to be suited well for handling uncertain data corresponding to a Gaussian distribution. To show that ordi-

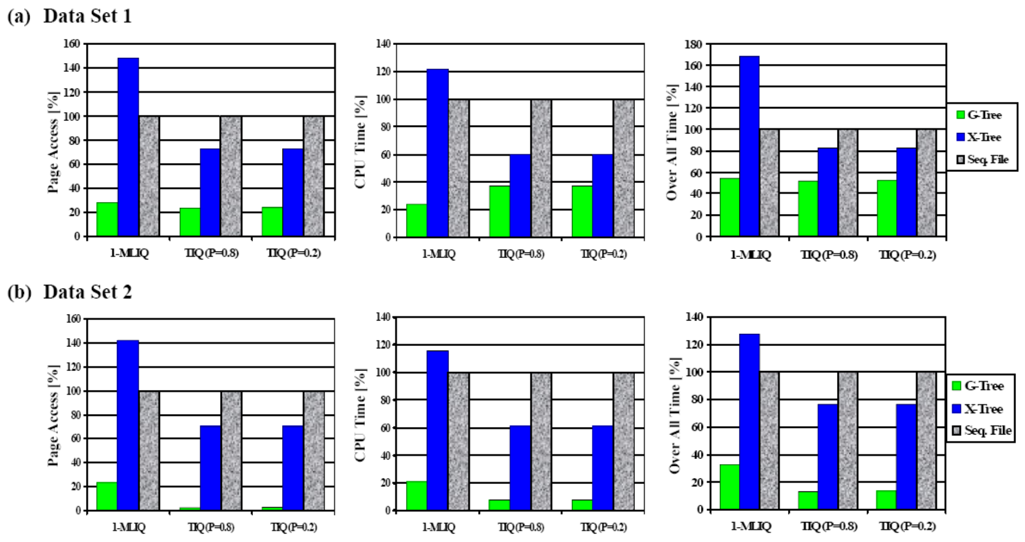


Figure 3.7: Performance of sequential scan, X-tree on hyper-rectangle approximations of pfv and Gauss-tree on dataset 1 and dataset 2 (see text for details).

nary similarity search cannot improve its performance by using larger result sets, we increased the number of retrieved objects for the nearest neighbor query which increases the recall but decreases the precision. For dataset 1 the recall did not significantly increase. Even for a result set being 9 times bigger than necessary the recall reached only a value of 60%. For dataset 2 the recall could be improved to 97% when using more than 6 times the size of the result set that is necessary. However, due to the dependency between precision and recall, the precision dropped to only 18%. Thus, the right selection of k cannot compensate for the missing handling of uncertainty.

In the next set of experiments, we compare the efficiency of query processing when using the Gauss-tree to the basic solution of a sequential scan over the complete database. To additionally compare to a more sophisticated method, we use an X-tree to store rectangular approximation of each pfv. To derive these approximations, we calculate the 95% quantiles in each dimension, i.e. we determine the interval around the mean value of a Gaussian that contains a random observation with a probability of 95%. By combining these intervals to a hyperrectangle, we generate a good approximation for each pfv. To process a MLIQ with this method we first calculate an approximation for the query pfv. Afterwards, we use the X-tree to determine a candidate set consisting of all approximations that intersect with the query approximation. To find out the final result the candidate set is refined by calculating the exact probabilities. Let us note that this method does not offer exact results with respect to Gaussian uncertainty model because the used approximations allow false dismissals. However, using this method, we observed a precision and recall that was only slightly below the values we observed for the Gauss-tree. Figure 3.7 displays the page accesses, cpu time and complete query time for an MLIQ and two TIQ ($P_\theta = 0.2$ and $P_\theta = 0.8$) on both datasets. All values are given in percent to the values of the sequential scan using the Gaussian uncertainty model. For dataset 1, the Gauss-tree was able to reduce the page accesses as well as the CPU time up by a factor of 4.2 compared to the sequential scan for all three query types. Though, the all over time suffered from additional seeks on the hard disc, the Gauss-tree was still able to improve query processing by at least 46% for all three types of queries. For dataset 2, the Gauss-tree achieved a speed up of 4.3 with respect to the number of accessed pages and of 4.8 for the cpu time of the MLIQ. For the TIQ, it even achieved to improve the page accesses by a factor between 35.7 and 43.2 of the page accesses of the sequential scan and the cpu time by a factor of 13.2 of the cpu time of the sequential scan. The speed up of the all over query time was between 3.1 for the MLIQ and about 7.5 for both TIQ. Thus, the Gauss-tree offered a significant improvement of the efficiency compared to the sequential scan. The X-tree storing rectangular approximations on the other hand did not offer any speed up against the sequential scan for MLIQ. Though it achieved some improvement for the TIQ, it was only capable to decrease the all over time of both queries by 17.3% for dataset 1 and 23.2% for dataset 2. Thus, it did not offer any real benefit.

3.7 Conclusions

In this chapter, we introduced the Gaussian uncertainty model for identification queries on inexact, probabilistic feature vectors (pfv). This model extends feature vectors by an additional uncertainty value for each dimension, associating each feature vector to a multivariate Gaussian distribution. Based on this model, we defined novel types of queries called k-most-likely identification queries (k-MLIQ) and threshold identification queries (TIQ). To speed up query types such as TIQ or k-MLIQ, we proposed the Gauss-tree, a balanced index structure from the R-tree family which does not index the Gaussian curves as spatial objects but instead the parameter space of the means and standard deviations of the Gaussians. We developed the algorithms for both insertion and split as well as query processing for TIQ and k-MLIQ. In our experimental evaluation, we demonstrated the superior quality of the query result when using probabilistic feature vectors as well as the efficiency when using the Gauss-tree.

Chapter 4

High Performance Video Retrieval using Probabilistic Feature Vectors

Content based multimedia retrieval is an important topic in advanced database systems. Specifically, an emerging and challenging task in this area is the content based search in video data. This chapter introduces a novel method for similarity search in video databases. It is organized as follows. Section 4.1 motivates similarity search in video databases. Afterwards, Section 4.2 surveys related topics like content based video retrieval and similarity search using point sets, and probabilistic feature vectors. In Section 4.3, we formalize our model and the new query types. In Section 4.4, we describe the new algorithms for processing the proposed query types. To demonstrate the quality of our approach to video retrieval and show the superior efficiency of our query algorithms, we provide several experiments on a database of over 900 video clips in Section 4.5. In Chapter 4.6, we propose ProVeR, a prototype search engine for content-based video retrieval which represents a video as a set of Gaussians. This chapter is concluded by Section 4.7.

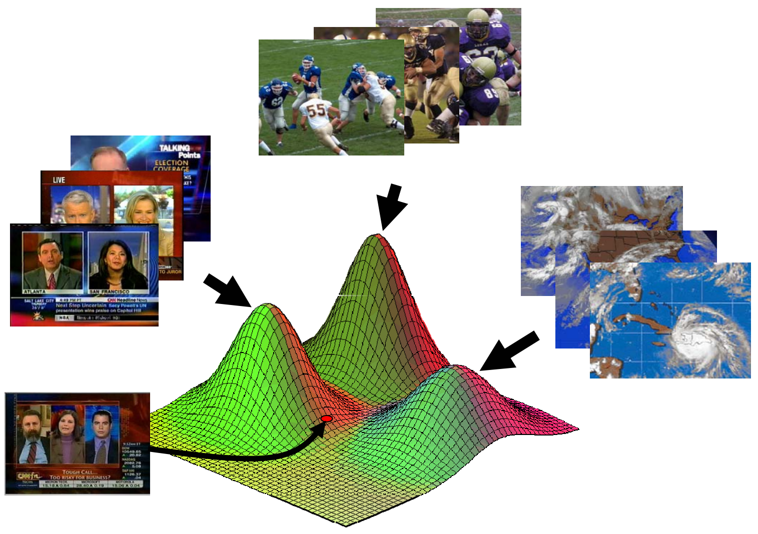


Figure 4.1: A news video clip summarized as set of probabilistic feature vectors.

4.1 Introduction

Video clips are an important type of multimedia data. Due to recent technical advances, the amount of video data that is available in digital formats as well as the possibility to access and display such video files has increased enormously. Nowadays, it is possible to view complete movies on mobile phones and MP3 players. Another important aspect is that broadcasting videos over the WWW (e.g. in video podcasts) allows to distribute video data to a large number of people while spending minimum effort and budget.

The enormous amount of video clips and movies that is currently available causes a need for database techniques to manage, store and retrieve video data for various applications. In this chapter, we focus on the following scenario: Given a database of movies or video clips, we want to retrieve all movies from the database that are likely to match a given set of query images. The query images might consist of a continuous image sequence of a scene or might be sampled from the complete movie. For this type of scenario, there are various applications. For example, a company wants to determine if a given video podcast or shared video file contains scenes from any copyright protected movie or video clip. In this scenario, the company would store all of its movies in the database and automatically check if the scenes in the video podcast match any scenes in the database.

Another example is a database of news programs recorded on various days from various tv stations. A user can retrieve all news programs that are likely to contain a given video clip featuring a particular event. Since most news programs use videos which are provided by video news agencies, it is very likely that the news programs dealing with similar topics contain similar news clips. Another application is the detection of commercials in video data recorded from television. In this case, the commercial is the query and the programs are stored in the database. Thus, there are varying applications for this scenario varying from the detection of single scenes to similarity search on complete movies.

From a technical point of view video data consists of a sequence of images (so-called frames) that might be accompanied with some soundtrack. In our approach, we focus on the image part only. To allow similarity search on video clips, each frame is usually represented by a feature vector. So-called summarization [ZRHM98, GGM02, CSL99b] techniques are used to reduce the enormous number of frames. For summarization, a video is decomposed into shots, i.e. a sequence of frames within a movie showing the same scenario recorded from the same camera position. The images within a shot are usually very similar and thus, the images are usually associated to very similar feature vectors. Therefore, each shot can be summarized by some representative object and only the representative objects are stored in the database. To represent a shot, it is often sufficient to simply take the centroid or mean vector of all feature vectors within the shot. Newer approaches like [IBW+04] represent shots as Gaussian probability density functions (pdf) where each component μ_i of the mean vector is complimented by a variance

σ_i^2. We call such feature vectors where each vector component is associated to a variance value *probabilistic feature vector* (pfv). This type of summarization is usually more accurate because the method additionally considers the variance among the summarized feature values. In our new approach, we condense the given video data even more, by representing all similar frames by one Gaussian regardless of the shot they belong to. To conclude, each movie in the database is represented by a *set of probabilistic feature vectors* (pfvs) where each Gaussians represents a set of similar frames.

Our work is focused on similarity search and scene detection in movie databases. To pose a query, a user has to provide a video clip that might comprise a scene in the movie or even the complete movie. The query clip can be transformed into a set of frames, corresponding to a set of traditional feature vectors or a set of probabilistic feature vectors. To use probabilistic (rather than traditional) feature vectors for the *queries* yields advantages as well as disadvantages: extracting a set of frames and determining traditional feature vectors without further summarization might be computationally simpler and less expensive. In contrast, probabilistic feature vectors might represent the information contained in the query in a more concise way. Therefore, we will examine both possibilities.

Furthermore, we develop a method for comparing both types of query representations to objects stored in the database which is based on the likelihood that the query matches the database object. Based on this method, we describe two types of probabilistic queries. The first type is the *set-valued probabilistic threshold query* retrieving all movies matching the given query frames with a likelihood which is higher than a specified threshold value. The second query type is the *set-valued probabilistic ranking query* retrieving the top k movies from the database which are most likely query hits.

Although summarization considerably decreases the size of the representation of each database object, query processing still requires to examine every movie description in the database. Therefore, we will introduce algorithms for query processing that are facilitated by the Gauss-tree in Chapter 3, an index structure for probabilistic feature vectors. However, previous work on the Gauss-tree was focused on querying single objects. In this chapter, we introduce techniques for querying set-valued objects which are more complex.

4.2 Related Work

Video Summarization Techniques. Since video data consists of large sequences of images or frames, a straightforward feature representation of a movie might contain thousands or even millions of feature vectors. In order to handle such data efficiently, summarization techniques are usually applied to the original data, i.e. the original feature vectors are grouped together and each group is represented by a summarization vector or summarization representative, cf. Section 2.1.5 in Chapter 2 for details. However, to the best of our knowledge, none of these techniques uses an index structure for the pfvs to accelerate query processing.

Similarity Search Based on Set-Valued Objects. Set-valued objects are usually compared by complex distance measures like [EM97, RB01] allowing similarity queries as discussed in Section 2.1.4 in Chapter 2. However, selecting a suitable distance measure for a particular application is often quite difficult because there exist many different notions of similarity between two sets of feature vectors. Another problem is the understandability of the derived distances. For complex distance measures and large set-valued objects containing hundreds of instances, it is very difficult to understand why the set-valued objects are similar. Finally, employing the proposed distance measures often yields efficiency problems. Since most of the distance measures for set-valued objects are non-metric, employing index structures is not always possible. Additionally, useful filter steps avoiding time consuming distance calculations like in [BKK+03] were introduced for a minority of multi-instance distance measures only. To the best of our knowledge there is so far no query algorithm handling sets of probabilistic feature vectors, instead of ordinary set-valued objects.

Similarity Search Based on Probabilistic Feature Vectors. In [CKP03] a new uncertainty model is introduced and several new types of queries are described that allow the handling of inexact data. [CXP+04] describes two methods for efficiently answering probabilistic threshold queries that are based on the R-Tree [Gut84]. A probabilistic threshold query returns all data objects that are placed in a given query interval with a probability exceeding a specified threshold value. [TCX+05] introduced the U-Tree for indexing uncertain 2D objects. All these approaches do not handle sets of probabilistic feature vectors and do not apply a Baysian setting. Thus, the mentioned approaches are rather dealing with data objects having an uncertain location.

4.3 Video Retrieval using Probabilistic Feature Vectors

In this section, we will formalize video summarization using sets of probabilistic feature vectors (pfvs) following a Gaussian density function. Additionally, we will provide the probabilistic framework for comparing queries to movies and specify the new types of queries.

As mentioned before, the video part of a movie is a sequence of images which can be transformed into d-dimensional feature vectors $f \in \mathbb{R}^d$. Applying summarization techniques, a video is represented by a set of pfvs. Let us note that there are other notions of pfvs which are based on different density function, but in this chapter the distribution function of a pfv is considered to be Gaussian. Thus, our pfvs are defined as proposed in definition 3.1.

To represent a movie, we employ a set of pfvs. Each pfv is considered to represent a set of similar frames in the movie. Additional to each pfv, we consider a weight expressing the average amount of frames represented by the given pfv in the complete movie. Thus, pfvs representing more frames have larger weights than pfvs representing a smaller fraction of the frames. We can now define a movie descriptor as follows:

Definition 4.1 (Movie Descriptor)
A movie descriptor M is a set of pfvs $\{v_1, \ldots, v_k\}$ and a weighting $\{w_1, \ldots, w_k\}$. w_i corresponds to the a priori likelihood that a frame in the movie is described by the pfv v_i. Furthermore, the following condition holds:

$$\sum_{i=1}^{k} w_i = 1.$$

A query is posed by specifying a video clip or only a part of it. To calculate the likelihood that the query is contained in some database object, we first of all have to apply some feature transformation to the query as well. Thus, a query Q can be considered as a set of feature vectors $\{q_1, \ldots, q_l\}$ with $q_i \in \mathbb{R}^d$. To calculate the probability that Q is contained in a movie described by M, we first of all have to derive a probability for a single query frame q_i for being contained in a given pfv $v_j \in M$ having the weight w_j. A pfv corresponds to a density function over $\mathbb{R}^d$. Thus, we can calculate the density of q_i w.r.t. v_i. However, to calculate a probability for a single vector in a continues space, we would have to integrate over some interval. Since for a single vector this interval converges to 0, the probability of the vector converges to 0 as well. However, since we already observed q_i, we actually do not need to calculate the probability that exactly q_i occurs in the given video. Instead, we can apply the theorem of Bayes and calculate the conditional probability that q_i belong to v_j under the condition it appeared at all. To formalize this condition, we have to distinguish three cases. First, q_i belongs indeed to v_j. Second, q_i belongs to some other pfv v_k in the same movie M. Finally, q_i is not contained in M but is part of some other movie. To approximate the last case, we specify $H_0(q_i)$ which is modeled by a uniform distribution or the average density of any known pfv for the vector q_i. Additionally, we multiply this density with the number of pfvs in the compared movie descriptor to have a weighting which is equal to the movie descriptor.

Thus, the probability that q_i appears at all is the sum of the probabilities $p(q_i|v_i)$ that q_i belongs to some v_i describing the current movie M and the probability that $q_i$is not contained in M which is formulated in $H_0(q_i)$. Formally, we can calculate the probability $P(v_j|q_i)$:

$$P(v_j|q_i) = \frac{w_j \cdot p(q_i|v_j)}{\sum_{\hat{v} \in V} \hat{w} \cdot p(q_i|\hat{v}) + H_0(q_i)}$$

Since a movie is given by a set of pfvs, the probability that a frame q_i is contained in the complete movie described by M, can be computed by summing up the probabilities for each pfv:

$$P(M|q_i) = \sum_{v_j \in M} P(v_j|q_i)$$

Finally, we have to consider all frames $q_i \in Q$ of a query. Thus, we calculate the average probability for any frame in the query q_i for being contained in the given movie descriptor M by:

$$P(M|Q) = \frac{\sum_{q \in Q} P(M|q)}{|Q|}$$

If a query comprises large numbers of frames this method yields performance problems. Thus, we have to reduce the number of frames for the query object as well. If the query must be answered in interactive time, sophisticated summarization techniques cannot be applied. Thus, we propose a simple reduction by considering every ith frame only. If time is less important, summarization by sets of pfvs is applicable. In this case, the query is represented by a movie descriptor itself. For calculating the probability that a movie descriptor M_q describes frames which are contained in the movie described by M, we will proceed as follows. We first of all determine the probability that a query pfv v_q describes the same set of feature vectors as a pfv v_m contained in the movie. This probability can be defined as follows:

The probability density of two Gaussians for describing the same vector can be specified as follows:

$$p(v_q, v_m) = \int_{-\infty}^{+\infty} p(v_q|x)p(v_m|x)dx$$

Having this probability, we can calculate the conditional probability for v_m under the condition of v_q in the following way:

$$P(v_m|v_q) = \frac{w_m \cdot w_q \cdot p(v_q, v_m)}{\sum_{\hat{v} \in M} \hat{w} \cdot w_q \cdot p(v_q, \hat{v}) + H_0}$$

Using this probability, we can proceed as above. The probability for $P(M|M_q)$ is the average probability of $P(M|v_q)$ which is the sum over all $P(v_j|v_q)$ in M:

$$P(M|M_q) = \frac{\sum_{v_q \in M_q} \sum_{v_j \in M} P(v_j|v_q)}{|Q|}$$

Based on these probabilities, we can specify probabilistic queries retrieving any movie in the database having a large enough probability for containing a query video clip. To decide which probability is large enough for being contained in the result set, there are two general approaches. The first is to define a fixed probability threshold, e.g. 80%. Thus, we retrieve all movies containing the specified query frames with a probability of more than 80%. Formally, we can define a set-valued probabilistic threshold query on movie descriptors as follows:

Definition 4.2 (Set-Valued Probabilistic Threshold Query) *(SVPTQ)*
Let DB be a database of movie descriptors, let Q be a set of query frames and let $P_\theta \in [0 \ldots 1]$ be a probability threshold. The answer of a threshold identification query is defined as follows:

$$SVPTQ_{DB}(Q, P_\theta) = \{M \in \mathcal{DB} | P(M|Q) \geq P_\theta\}$$

The second method for deciding containment in the query result is to retrieve the k most likely results. Thus, the threshold is relative to the database content. An example for this type of query is: Retrieve the 5 movies from the database having the highest probability for containing the query scene. We will call this type of query set-valued probabilistic ranking query (SVPRQ). In the following we will formalize SVPRQs:

Definition 4.3 (Set-Valued Probabilistic Ranking Query) *(SVPRQ)*
Let DB be a database of movie descriptors M, let Q be a set of query frames and let $k \in \mathbb{N}$ be a natural number. Then, the answer to a set-valued probabilistic ranking query (SVPRQ) on DB is defined as the smallest set $RQ_k(Q) \subseteq \mathcal{DB}$ with at least k elements fulfilling the following condition:

$$\forall M_a \in RQ_k(Q), \forall M_{db} \in \mathcal{DB} \setminus RQ_k(Q) : P(M_a|Q) > P(M_{db}|Q)$$

4.4 Indexing Summarized Videos

After describing the queries, we are now introducing our solution for efficient query processing based on sets of probabilistic feature vectors.

4.4.1 Answering Set-Valued Queries

In contrast to searching in a database where each object is represented by a single pfv, our application requires the use of set-valued objects for both the query and the database objects. For query processing, we have to match all the elements of the query representation (being traditional or probabilistic feature vectors) against all the movie descriptors in the database. The difficulty of this task lies in the problem that even if a movie descriptor offers a high likelihood for containing one of the elements of our query, the corresponding movie needs not necessarily to be a likely candidate for containing the complete query. Thus, in order to prune a movie descriptor from the search space, it is necessary to approximate the probability of the complete movie descriptor for matching the complete query.

Our new method for indexing movie descriptors uses a single Gauss-tree for managing all pfvs belonging to any movie descriptor in the database. Each pfv is identified by its movie ID and an additional sequence number identifying the pfv within the movie. To utilize this data structure for answering matching queries, we will describe conservative approximations of the likelihood that the elements of a query Q are described by some movie descriptor being stored in a set of nodes belonging to the Gauss-tree.

Therefore, we will first of all calculate the probability of a query element $q_i \in Q$ that q_i is contained in some movie M descriptor which is completely stored in a set of nodes P:

Lemma 4.1 *Let Q be a set valued query, let $P = \{p_1, \ldots, p_m\}$ be a set of nodes in the Gauss-tree T containing the pfvs of a movie Descriptor $M \in \mathcal{DB}$. We define the function $maxDense_P(q)$ as follows:*

$$maxDense_P(q) = \max_{p_i \in P} N_{p_i}(q)$$

Then the following condition holds for all $q \in Q$:

$$\forall M \in P : P(M|q) \leq \frac{maxDense_P(q)}{maxDense_P(q) + H_0}$$

Proof.

$$P(M|q) = \frac{\sum_{v_i \in M} w_i \cdot p(q|v)}{\sum_{v_i \in M} w_i \cdot p(q|v) + H_0(q)} \leq \frac{\max_{p_j \in P} N_{p_j}(q)}{\max_{p_j \in P} N_{p_j}(q) + H_0(q)}$$
$$\Leftrightarrow \sum_{v_i \in M} w_i \cdot p(q|v) \leq \max_{p_j \in P} N_{p_j}(q)$$
$$\Leftrightarrow \sum_{v_i \in M} w_i \cdot p(q|v) \leq \sum_{v_i \in M} w_i \cdot \max_{p_j \in P} N_{p_j}(q)$$
$$= \max_{p_j \in P} N_{p_j}(q) \cdot \sum_{v_i \in M} w_i = \max_{p_j \in P} N_{p_j}(q) \cdot 1$$

$\square$

Based on this lemma, we can determine the maximum probability for each element q of the query Q of being contained in a movie M which is completely stored in the set of pages P. To employ this lemma for approximating the likelihood of the complete query Q, we must take the average of the conservative approximations over all elements of the query Q. The average of a set of conservative approximations must be a conservative approximation of the average of the exact values. Since each part of the sum in the average of approximations is greater or equal to the exact value, the sum of approximations is greater or equal than the sum of exact values as well. The average is the mentioned sum divided by the number of elements. Therefore, the following condition holds:

$$\forall M \in P : P(M|Q) \leq \frac{1}{|Q|} \cdot \sum_{q_i \in Q} \frac{maxDense_P(q)}{maxDense_P(q) + H_0(q)}$$

Though we can now approximate the probability that Q matches some movie $M \in P$, the approximation is potentially depending on several nodes

$p \in P$ at the same time. For ranking and pruning nodes in the query algorithms, we therefore prove the following lemma:

Lemma 4.2 *Let Q be a set-valued query, let $P = \{p_1, \ldots, p_m\}$ be a set of nodes in the Gauss-tree T containing the pfvs of any movie descriptor $M \in \mathcal{DB}$. Then the following condition holds:*

$$\begin{aligned}\forall M \in P : P(M|Q) &\leq \max_{p \in P, q \in Q} \frac{N_p(q)}{N_p(q) + H_0(q)} \\ &= \max_{p \in P} maxProb(Q, n)\end{aligned}$$

Proof.

$$\begin{aligned}\forall M \in P : P(M|Q) &\leq \frac{1}{|Q|} \cdot \sum_{q \in Q} \frac{\max_{p \in P} N_p(q)}{\max_{p \in P} N_p(q) + H_0(q)} \\ &\leq \frac{|Q|}{|Q|} \cdot \max_{q_i \in Q} \frac{\max_{p \in P} N_p(q)}{\max_{p \in P} N_p(q) + H_0(q)} \\ = \max_{q \in Q} \max_{p \in P} \frac{N_p(q)}{N_p(q) + H_0(q)} &= \max_{p \in P, q \in Q} \frac{N_p(q)}{N_p(q) + H_0(q)}\end{aligned}$$

□

We can now approximate the probability $P(M|Q)$ that M is completely stored in the set of nodes P on the basis of a single node p_{max} where p_{max} is the node p maximizing $maxProb(Q, p)$. An important property of this approximation is that it can be used to rank the access order of the nodes in the Gauss-tree for query processing. Additionally, we will employ this lemma for pruning unnecessary pages and terminate our queries.

Our algorithms employ two data structures. The first is a priority queue containing the nodes of the Gauss-tree that have not been examined yet. The priority is ranked with respect to $maxProb(Q, p)$ in descending order. Due to Lemma 4.2, $maxProb(Q, p)$ yields an upper bound of the probability of a movie descriptor to be completely stored in the remaining nodes of the tree.

Additionally, $maxProb(Q, p)$ can be considered as the maximum probability for all query elements that are yet unknown.

The above lemmas describe the case that there is a set of the nodes which are guaranteed to contain the complete set of considered movie descriptors. However, during query processing we will encounter the case that we already retrieved some pfvs for a movie M, but there are still some $v \in M$ which are stored in the part of the Gauss-tree that has not been examined yet. For those movie descriptors, we have to store the already known densities in the so-called candidate table until the complete set of pfvs is retrieved. Each entry in the candidate table corresponds to a movie descriptor. For each movie stored in the candidate table, we additionally store the sum of the densities for each query element q and each density function v_i that has been retrieved so far. Let us note that each density $p(q|v_i)$ in each sum is weighted with w_i which is the weight of the pfv v_i in the descriptor M. Finally, we store the number of all already retrieved density functions for each movie descriptor M. Based on this data and the current $maxProb(Q, p)$ on the top of our priority queue, we can also approximate the density of any partly known movie descriptor. The approximation is formulated in the following lemma:

Lemma 4.3 *Let M be a partially retrieved movie descriptor, $A \subset M$ be the set of already known pfvs with weight w_a and let $B \subset M$ be the still unknown elements of M. Furthermore, let P be the set of node in the Gauss-tree P containing B. We define the function $partDensity_A(q)$ as follows:*

$$partDensity_A(q) = \sum_{v_i \in A} w_i \cdot p(q|v_i) + (1 - \sum_{v_i \in A} w_i) \cdot maxDense_P(q)$$

Then, the following condition holds:

$$P(M|q) \leq \frac{partDensity_A(q)}{partDensity_A(q) + H_0(q)}$$

Furthermore, we can state for the complete query Q:

$$P(M|Q) \leq \frac{1}{|Q|} \cdot \sum_{q \in Q} \frac{partDensity_A(q)}{partDensity_A(q) + H_0(q)}$$

Proof. *The proof is analogue to the proof of lemma 4.2.* □

4.4.2 Set-Valued Probabilistic Threshold Query

In our first query, we have a fixed global probability threshold P_Θ which can be employed to decide whether a movie is part of the result set. We will now explain our algorithm for processing SVPTQs using the Gauss-tree. The pseudocode of this algorithm is displayed in Figure 4.2. The algorithm starts by reading the root node of the Gauss-tree. For each node p being a child node of the root, we now calculate $maxProb(Q, p)$ and insert the nodes into the priority queue which is sorted in descending order. Afterwards, the algorithm enters its main loop which iterates until the priority queue is empty. Additionally, the algorithm terminates if we can guarantee that there cannot be any movie descriptor left matching the given query Q with a likelihood larger than P_Θ. In each step, the algorithm removes the top element of the queue. If the element is a node, it is loaded and pointers to its child nodes are inserted into the priority queue, ranked by $maxProb(Q, p)$. If the top element of the queue is a pfv, we check if there is already an entry in the candidate table corresponding to the movie descriptor M of the pfv. If not, we insert a new entry into the candidate table. In both cases, we can update the sum for each query element for the movie descriptors in the candidate table. If the current entry for the movie descriptor M is complete, i.e. all of its pfvs have been retrieved, we can calculate the likelihood. If this likelihood is larger than t, we can add M to the result set. Finally, the entry for M is removed from the candidate table.

If the movie descriptor M is not complete after updating the priority

```
SVPTQ(Query Q, float $P_\Theta$)
  activePages := new PriorityQueue(descending)
  candidateTable := new CandidateTable()
  result := new List()
  pruned := new List()
  $activePagesQueue$.insert(root, 1.0)
  DO
    $aktNode$ = activePages.removeFirst()
    IF $aktNode$ is a directory node THEN
      FOR each $node$ in $aktNode$ DO
        activePages.insert($node$,$maxProb(Q, node)$)
      END FOR
    END IF
    IF $aktNode$ is a data node THEN
      FOR each $pfv$ in $aktNode$ DO
        IF pfv.MovieID in pruned THEN
          CONTINUE
        END IF
        candidateTable.update(pfv.MovieID, pfv(Q))
        candidateEntry := candidateTable.get(pfv.MovieID)
        IF candidateEntry.isComplete THEN
          IF candidateEntry.probability(Q) $\geq P_\Theta$ THEN
            result.add(pfv.MovieID)
          END IF
          candidateTable.delete(pfv.MovieID)
        ELSE
          IF andidateEntry.approximation(Q) $\leq P_\Theta$ THEN
            pruned.add(pfv.MovieID)
            candidateTable.delete(pfv.MovieID)
          END IF
        END IF
      END FOR
    END IF
  WHILE((not candidateTable.isEmpty
     or activePages.topProbability $> P_\Theta$)
    and not activePages.isEmpty())
  RETURN $result$;
```

Figure 4.2: Pseudocode of set-valued probabilistic threshold query (SVPTQ).

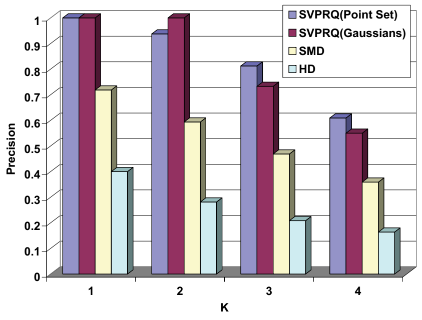
queue, we approximate the current maximum likelihood of M and Q. If the conservative approximation is smaller than t, we can exclude M from the result set. Thus, we store the ID of M in a separated pruning list and delete its entry from the candidate table. If we later encounter a pfv belonging to M, we can safely skip its computation after checking the pruning list. Our algorithm terminates if $maxProb(Q, p)$ for the top element of the priority is smaller than P_Θ. Additionally, we have to continue processing until the candidate table is empty, to make sure that the result is complete.

4.4.3 Set-Valued Probabilistic Ranking Query

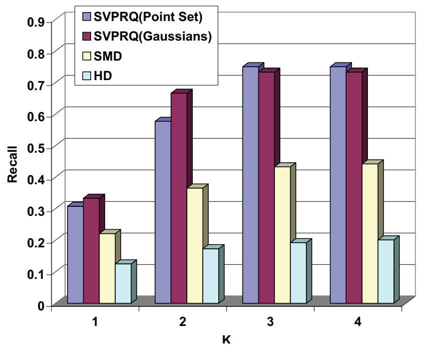
The second query type proposed in this chapter are SVPRQs. For SVPRs the minimum probability for a result depends on the movie having the k highest probabilities for containing the query set. The idea of the algorithm is quite similar to the previous algorithm. However, for this type of query, we need a second priority queue storing those k movies which currently have the largest probabilities for containing Q. We will sort this second priority queue in ascending order and refer to it as result queue. The pseudocode for this algorithm is displayed in Figure 4.3. We start again by ordering the descendant nodes of the root page w.r.t. $maxProb(Q, p)$. Afterwards we enter the main loop of the algorithm and remove the top element of the queue. If this element is a node, we load its child nodes. If these child nodes are nodes themselves, we determine $maxProb(Q, p)$ and update the priority queue. If the child nodes are pfvs, we check the candidate table for corresponding movie descriptor M and insert a new descriptor, in the case that there is not already a descriptor for the movie M. Afterwards, we can update the candidate table as mentioned before. If a movie descriptor M has been read completely, we can delete it from the candidate table and compare its probability $P(M|Q)$ to the probability of the top element of the result queue, i.e. the movie descriptor encountered so far having the k

```
SVPRQ(Query Q, integer $k$)
  activePages := new PriorityQueue(descending)
  resultQueue := new PriorityQueue(ascending)
  candidateTable := new CandidateTable()
  pruned := new List()
  $activePagesQueue$.insert(root, 1.0)
  DO
    $aktNode$ = activePages.removeFirst()
    IF $aktNode$ is a directory node THEN
      FOR each $node$ in $aktNode$ DO
        activePages.insert($node$,$maxProb(Q, node)$)
      END FOR
    END IF
    IF $aktNode$ is a data node THEN
      FOR each $pfv$ in $aktNode$ DO
        IF pfv.MovieID in pruned THEN
          CONTINUE
        END IF
        candidateTable.update(pfv.MovieID, pfv(Q))
        candidateEntry := candidateTable.get(pfv.MovieID)
        IF candidateEntry.isComplete THEN
        prob := candidateEntry.probability(Q)
          IF prob$\geq$ resultQueue.topProbability THEN
            IF resultQueue.size = $k$ THEN
              resultQueue.removeFirst
            END IF
            resultQueue.add(pfv.MovieID,prob)
          END IF
          candidateTable.delete(pfv.MovieID)
        ELSE
          IF candidateEntry.approximation(Q) $\leq$
            resultqueue.topProbability THEN
            pruned.add(pfv.MovieID)
            candidateTable.delete(pfv.MovieID)
          END IF
        END IF
      END FOR
    END IF
  WHILE((not candidateTable.isEmpty
    or activePages.topProbability > resultqueue.topProbability)
    and not activePages.isEmpty())
  RETURN $result$;
```

Figure 4.3: Pseudocode of set-valued probabilistic ranking query (SVPRQ).



(a) Precision.



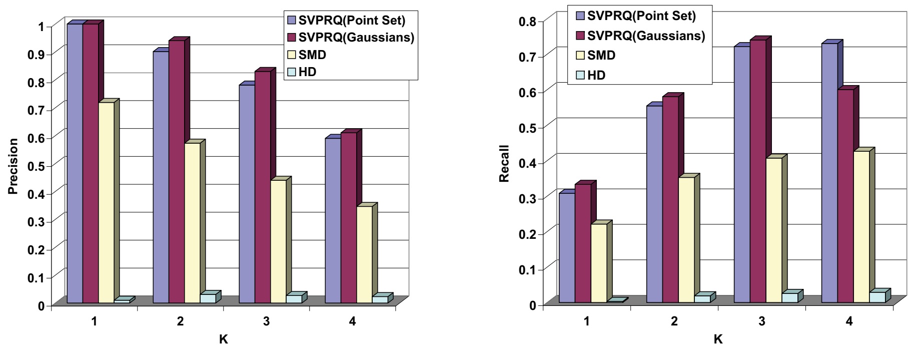
(b) Recall.

Figure 4.4: Precision and recall achieved on similarity search by SVPRQ and its comparison partners on complete video retrieval.

highest probability. If the probability of M is higher than that of the top element, we need to add M to the queue. However, to make sure that we do not retrieve more than k elements, we have to check the size of the result queue. If there are already k elements, we have to remove the top element before inserting M. In the case, that the entry in the candidate table does not contain the complete information about M yet, we still can calculate a probability estimation and compare it to the top element of the result queue. If $P(M|Q)$ is smaller than the k highest probability in the result queue, we can guarantee that M is not a potential result. Thus, M is deleted from the candidate table and stored in our list for excluded movie descriptors. The algorithm terminates if the top of the priority containing the remaining notes provides a lower value than the top of the result queue and the candidate table is empty.

4.5 Experimental Evaluation

Testbed. All experiments were performed on a workstation featuring a 2.2 GHz Opteron CPU and 8GB RAM. All algorithms are implemented in



(a) Precision. **(b)** Recall.

Figure 4.5: Precision and recall achieved on similarity search by SVPRQ and its comparison partners using scene retrieval.

Java 5. We evaluated our SVPTQ, SVPRQ and their comparison partner using a database of 902 music video clips recorded from various TV stations. The average length of a video clip within our collection is 4 minutes and 14 seconds. We extracted the image representations of the videos on a per-frame basis, i.e. we generated 25 features/second for PAL and 30 features/second for NTSC videos. From each image, we extracted a color histogram. For the color histogram, we used the HSV color space which was divided into 32 subspaces, 8 ranges of hue and 4 ranges of saturation.

In order to obtain the summarization for each video clip, we applied the EM clustering algorithm. The EM clustering provided us with approximately 100 multivariate Gaussians per video clip. In our experiments, we performed video similarity search. As setup step, we picked 40 query videos from our database and manually selected a set of videos which are similar to the query videos.

To generate queries, we employed two methods for collecting query frames. The first method tried to capture the complete video clip. Thus, we sampled every 50th frame from the complete clip to derive a representative sample of frames. The second method simulated queries which are posed by giving

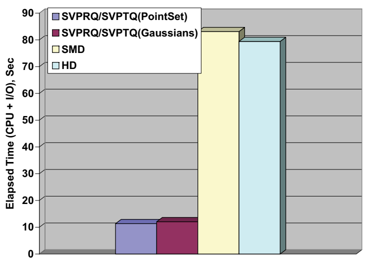


Figure 4.6: Elapsed average query time for SVPRQs and SVPTQs for the query on the complete video clips.

only a scene or shot from the video. Therefore, we sampled a random interval from the sequence of all frames in the video corresponding to about 500 frames, i.e. 20 seconds. For this type of query, we used every 10th frame of the query interval, i.e. we used 50 frames per query. Additional to these queries, we also generated queries which are represented by sets of probabilistic feature vectors. For representing the complete video, we again employed EM clustering for 100 clusters on the complete set of frames in one video clip. For the queries on the scenes, we clustered the 500 frames, deriving 5 Gaussians.

To have comparison partners for retrieving videos on sets of ordinary feature vectors, we generated a database containing color histograms for all frames of every video clip in our test set. We employed two well-established distance measures for set-valued objects to pose queries to this database, the Hausdorff (HD) distance and the sum of minimum distances (SMD)[EM97]. For these methods we could only use the query consisting of sets of feature vectors.

Our first set of experiments examined the precision and recall of video retrieval for all 4 types of generated queries. Therefore, we performed kNN queries for our comparison partners and SVPRQ for the methods proposed in this chapter. The result for the queries on the complete video clips is displayed in Figure 4.4. As a first result it can be seen that our new method significantly outperformed the compared methods w.r.t. precision and recall. For $k = 1$, we should retrieve the database object from which the query was generated, we achieved a precision of almost 1.0. For the 2nd nearest neighbor our method still achieved a precision of about 0.9 which is about 40% better than the best of our comparison partners (SMD). The chart displaying the recall of our query results displays a similar picture. The recall of our new methods considerably outperformed the compared methods. Furthermore, we achieved a recall of over 70 % for $k = 3$ which is the average number of similar videos for a query object in our test bed.

The experiments on the queries on parts of video clips display similar results. Our methods outperformed the compared method w.r.t both precision and recall. Though the performance advantage w.r.t. precision was smaller than in the previous experiment, our proposed method still managed to outperform the best comparison partner, SMD, by more than 20% for all values of k. The results w.r.t. recall display similar improvements as well. To conclude, representing video clips as sets of Gaussians is well suited for accurate video retrieval and outperforms method based on sets of feature vectors w.r.t. precision and recall.

For measuring the efficiency of our new methods for query processing, we recorded the time taken for processing all 40 queries representing the complete movie. For each query object, we performed several queries corresponding to several parameter setting ($1 < k < 7$ and $0.1 < P_\Theta < 0.7$). The results are displayed in Figure 4.6. The average query time for our new methods was approximately 7 times smaller than that of the compared methods. Additionally, it can be seen that using sets of probabilistic feature vectors as query representation did not cause a considerable longer average query time. Let us note that the time for generating the Gaussians of the query was not added to the query time. To conclude our new query algorithm considerably outperformed the compared methods w.r.t. efficiency as well.

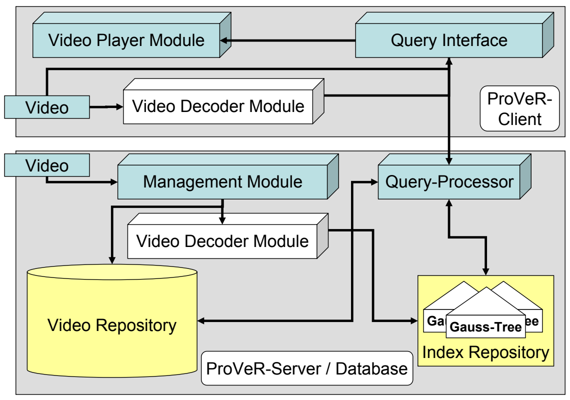


Figure 4.7: Architecture of a prototype video search engine.

4.6 ProVeR: Probabilistic Video Retrieval using the Gauss-Tree

4.6.1 System Architecture and Implementation

In order to demonstrate practical benefits of modeling objects by probability density functions, we propose a prototype search engine called ProVeR for content-based video retrieval which represents a video as a set of Gaussians. Figure 4.7 illustrates the client/server architecture of ProVeR. ProVeR provides even non-expert users with an intuitive method for efficient, content-based retrieval of videos containing similar shots and scenes. The server manages a video repository that contains video data that can be queried by the clients. Whenever a video is added to the repository by the management module, the video decoder module computes a summarization in form of a set of Gaussians. This step is performed for several different features, like e.g. color histograms. To support efficient query processing, each summarization for each representation is stored in a separate Gauss-tree. During query time, the user can choose between these feature representations. The server also manages a list of clients that are connected to its query processor module. A client processes query videos given by a user. For each query

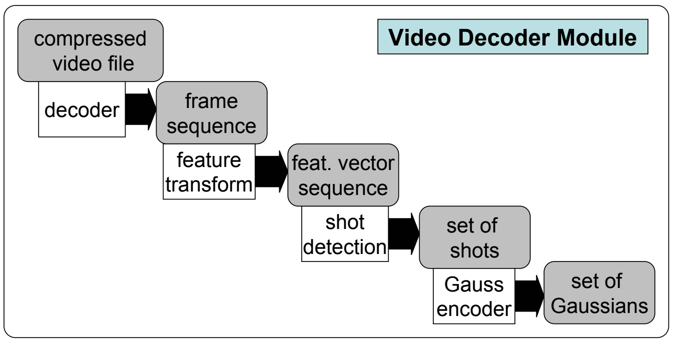
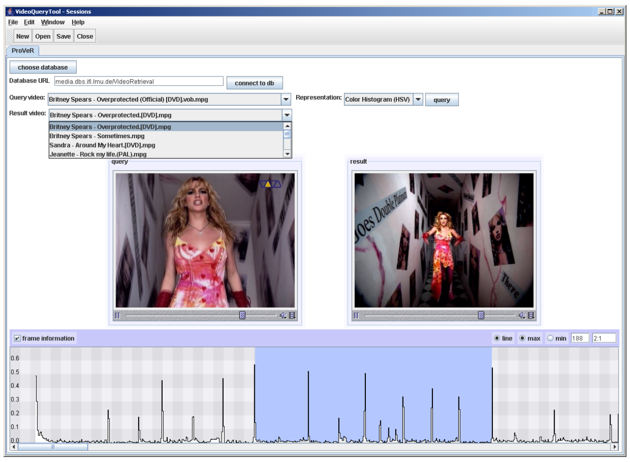


Figure 4.8: Video Decoder.

video, a summarization is generated by the client-side video decoder module. The calculated summarization is sent to the server which returns references to the k most likely videos in the repository. The videos in the result can be viewed by the video player module of the client.

An important part for the server as well as the client is the video decoder module. It generates a set of Gaussians for a given input video, cf. Figure 4.8. The output is computed in 4 steps. First, the video is decoded into a sequence of single images. While decoding, feature vectors for several different image representations are calculated. In the next step, a shot detection is performed for each representation. The sequence of feature vectors which corresponds to a single shot is then aggregated by a Gaussian. Let us note that the Gaussians for all representations can be simultaneously generated. Thus, the input video is described by a set of Gaussians.

ProVeR is implemented in Java 5.0 and runs on all platforms supporting the current version of the Java Runtime Environment. The feature extraction builds on the highly flexible Java Media Framework (JMF) 2.1, which provides easy access to a variety of video formats, e.g. MPEG-1. The videos are associated to several different representations. We extract color histograms, color moments and texture descriptions on a per-frame basis. The color histograms use the HSV color space which is divided into 32 subspaces, 8



Database URL: media.dbs.ifi.lmu.de/VideoRetrieval

Query video: Britney Spears - Overprotected (Official) [DVD].vob.mpg

Representation: Color Histogram (HSV)

Result video: Britney Spears - Overprotected.[DVD].mpg

Britney Spears - Overprotected.[DVD].mpg
Britney Spears - Sometimes.mpg
Sandra - Around My Heart.[DVD].mpg
Jeanette - Rock my life.(PAL).mpg

☑ frame information ● line ● max ○ min 188 2:1

Figure 4.9: Screenshot of ProVeR.

ranges of hue and 4 ranges of saturation. Additionally, we compute the color moments for the HSV color space. To capture the structural nature of the images, we also calculate the Haralick texture features [HSD73].

4.6.2 Practical Benefits

The ProVeR client starts with a list of known multimedia servers. Initially, the user chooses one of the available servers from the list. The client establishes a connection to the multimedia database on the server. In order to perform a query, the user has to supply a video file. While decoding the video, ProVeR extracts the image representations mentioned in Section 4.6. Depending on the selected representation, the client displays a distance graph in the bottom of the window (cf. Figure 4.9), providing information about the shot structure of the video. A valley in the distance graph indicates a sequence of similar images, which usually form a short. To specify a query, the user selects either a single shot or a sequence of subsequent shots. To display the content of the query selected query frames, ProVeR offers a preview consisting of a frame for each shot. For each selected shot, the client transforms the corresponding sequence of feature vectors into a single Gaussian. Thus, the sequence of selected shots is summarized by a set of Gaussians. This set is sent to the server as a query. Since the client sends only aggregated information to the server, the user doesn't have to share the original video data. Besides, this helps to save a lot of transmission bandwidth. Additionally, the decentralized approach also saves CPU time on the server. The server processes the query and returns a list of video repository references which contains the k most likely videos corresponding to the query. The user can browse through this result list and play a video file by selecting it, in which case it is streamed to the client.

4.7 Conclusions

In this chapter, we proposed efficient techniques for high performance video retrieval. Our methods are based on a summarization technique using probabilistic feature vectors, i.e. Gaussian probability density functions. For storage and efficient retrieval of probabilistic feature vectors, a specialized index structure, the Gauss-tree, was applied. Every video clip in the database is associated to a set of probabilistic feature vectors. A query video clip is also transformed into either a set of conventional feature vectors or into a set of probabilistic feature vectors. In both cases, query processing involves matching of sets of vectors. We defined two kinds of set-valued queries, set-valued probabilistic ranking queries and set-valued probabilistic threshold queries, and proposed efficient algorithms for query evaluation on top of the Gauss-tree. Our experimental evaluation using over 900 music video clips demonstrated the superiority of our approach with respect to both accuracy as well as efficiency of retrieval. In addition, we proposed ProVeR — a search engine for content-based video retrieval that offers an intuitive access to the shots and scenes contained in large video repositories. To allow efficient and effective retrieval, ProVeR represents shots as Gaussians which are stored in several Gauss-trees, one for each representation.

Chapter 5

Probabilistic Ranking Queries for Spatial Database Systems

In many advanced applications, there are no exact values available to describe the data objects. Instead, the feature values are considered to be uncertain. This uncertainty is modeled by probability distributions instead of exact feature values. A typical application of such an uncertainty model are moving objects where the position of each spatial object can be described by a bivariate normal distribution as illustrated in Figure 5.1. This chapter begins with an introduction into spatially uncertain objects in Section 5.1. Section 5.2 contains a brief description of related work in the area of indexing uncertain objects. In Section 5.3, we define the spatial Gaussian uncertainty model and two probabilistic query types for spatially uncertain data. Two novel query algorithms on the top of a Gauss-tree are discussed in Section 5.4. In our experimental evaluation in Section 5.6, we demonstrate that the Gauss-tree outperforms previously introduced query processing methods that are applicable to the spatial Gaussian uncertainty model. Finally, Section 5.7 concludes the chapter with a summary.

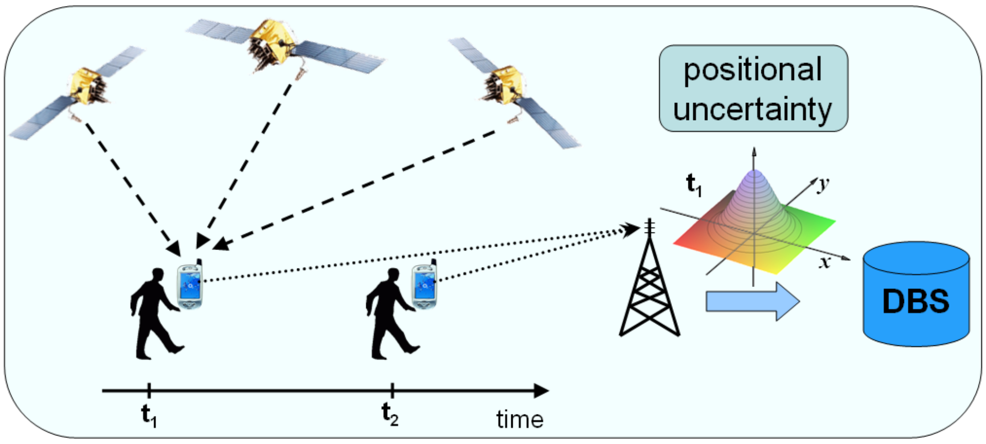


Figure 5.1: Spatial uncertainty in moving object database systems.

5.1 Introduction

In order to manage spatially uncertain objects in a database, an uncertainty model is needed to derive a probability distribution from the last observed feature values. A common approach which is described in [CKP03] is to assume that there is at least a certain interval where it can be guaranteed that the current value of the data object is contained in. Within this interval an arbitrary density distribution function is specified. We will refer to this approach as the interval uncertainty model. Though there exists a large variety of probability density functions, most applications rely on standard distributions like the uniform distribution or the Gaussian distribution for each data object. A disadvantage of the interval uncertainty model is the need to specify an interval which must contain the current object value. Though it is quite often possible to make some worst-case estimation, the resulting intervals often tend to be crude approximations of the current value which might be a problem for the selectivity of query processing. A solution to this approach is the use of distribution functions like the Gaussian where it is not necessary to specify an explicit interval. Since the density of a Gaussian rapidly decreases after a given distance to the mean value is reached, the area for which it is likely that the current object value is contained in, is limited in a natural way.

In this chapter, we therefore introduce another uncertainty model for spatial and sensor data, called Gaussian uncertainty model. The Gaussian density distribution is one of most established ways to describe uncertainty in a variety of applications. A Gaussian is defined w.r.t. two parameters, the mean value and the standard deviation. For example, to model the change of temperature, recorded by a sensor in a sensor network, the mean value can be assumed at the last observed exact value and the variance value can be estimated based on recent variations of the observed temperatures. For moving objects, applying the Gaussian uncertainty model is applicable as well (cf. Section 1.1 in Chapter 1 for further details). The mean vector corresponds to the position of the GPS antenna and the variance vector to positional uncertainty caused by measurement error (see Figure 5.1).

To conclude, the Gaussian uncertainty model relies on the Gaussian distribution to model the uncertain values only and is not bound to find an interval that is guaranteed to contain the actual current value. An important advantage of the Gaussian uncertainty model is that each object value is only complemented with one additional uncertainty attribute. Employing other distributions having p additional parameters increases the size of the database p times as well. This is a problem if we already assume limited storage capacity and bandwidth. Based on the Gaussian uncertainty model, we will discuss two important types of queries, probabilistic threshold queries (PTQs) defined in [CXP+04] and probabilistic ranking queries (PRQs). The second type of queries, the PRQs, has not been studied in context of spatial and sensor data yet. A PRQ retrieves those k objects which have the highest probabilistic of being located inside a given query area. To speed up processing these queries, we propose using the Gauss-tree(cf. Chapter 3). Based on the Gauss-tree, we describe algorithms for answering PRQs and PTQs.

5.2 Related Work

The Gauss-tree is a member of the R-Tree family which is a spatial index structure for indexing high dimensional data. The Gauss-tree was introduced in Chapter 3 to answer so-called identification queries which are based on a Bayesian uncertainty model that cannot be used for spatial uncertainty as discussed in this chapter.

As discussed in Section 2.1.2 of Chapter 2, several new techniques have been proposed in last years for handling uncertainty of spatiotemporal objects, e.g. method described in [CXP+04, TCX+05]. The differences of the approach introduced in [CXP+04] to our new approach are the following. The method described in [CXP+04] relies on a table to approximate the properties of one type of distribution function. Our method is based on the Gauss-tree introduced in Chapter 3. The Gauss-tree is for Gaussians only and thus, directly employs the Gaussian density function. In [CXP+04] a table is used to derive x-bounds for a given node in an index structure. For Gaussians, this parameter is equivalent to the standard deviation. Though the method were not originally designed for this purpose the method is adaptable to answering probabilistic ranking queries on the Gaussian uncertainty model as well. Our method calculates directly the maximum probability for any Gaussian in a data node for any given query interval. Unlike the method in [CXP+04] the Gauss-tree has its own split heuristic incorporating the non-linear characteristic of the standard deviation. [CXP+04] exclusively deals with probabilistic threshold queries and not with probabilistic ranking queries as we do in this chapter. [TCX+05] introduced the U-Tree for indexing uncertain 2D objects (cf. Section 2.1.2 for details). This method is not applicable to the Gaussian uncertainty model, because the planes start on the edges of the MBR. Thus, since we do not have any guaranteed area in the Gaussian uncertainty model. [DYM+05] introduces existential uncertainty. Though this method handles uncertainty as well, the methods for query processing cannot be applied to the problems discussed in this chapter.

5.3 Spatial Uncertainty Model and Query Types

In this section, we formalize notion of spatial uncertainty. Afterwards, we adapt the existing probabilistic threshold queries (PTQ) to our probabilistic model and define a novel query type — probabilistic ranking query (PRQ).

5.3.1 Gaussian Uncertainty Model

An uncertain data object v is described by d uncertain attribute values v_i with $1 \leq i \leq d$. For each uncertain attribute v_i, we cannot store an exact feature value, but store a probability density function describing the likelihood of all possible attribute values. In the Gaussian uncertainty model, we consider this density distribution function to be a Gaussian which is defined in Chapter 3, Definition 3.1.

To calculate the probability that an uncertain attribute value is contained in a certain query interval, we can integrate the Gaussian density function on the query interval.

Definition 5.1 (Gaussian Interval Probability)
For $a < b$ with $a, b \in \mathbb{R}$ the Gaussian probability for a given mean value μ and a standard deviation σ can be defined as follows:

$$P_{\mu,\sigma}(a,b) = Pr(v \in [a,b], \mu, \sigma) = \int_a^b N_{\mu,\sigma}(x)\ dx.$$

An object having d uncertain attributes which are specified by a vector of mean values $\vec{\mu}$ and a vector of standard deviations $\vec{\sigma}$ is called probabilistic feature vector (pfv) as defined in Chapter 3, Definition 3.1. For this pfv, we can calculate the probability that each attribute value v_i is contained in an attribute specific query interval $[a_i, b_i]$. Under the common assumption of attribute independency, calculating this probability can be done as follows:

$$Pr(v_i \in [a_i, b_i], \mu_i, \sigma_i, \forall i : 1 \leq i \leq d) = \prod_{i=1}^{d} P_{\mu_i,\sigma_i}(a_i, b_i).$$

5.3.2 Spatial Queries on the Gaussian Uncertainty Model

After describing a method to model uncertain data objects using Gaussians, we will now formally define two important types of queries on uncertain data objects. The first is the probabilistic threshold query (PTQ) which was first defined in [CKP03] for the interval uncertainty model. A PTQ computes all uncertain data objects that might be contained in a given query interval with a probability exceeding a given query threshold. For example, we want to retrieve all ships, that are likely to be found in a certain area of the ocean with a probability of at least 75%. Formally, a PTQ can be defined as follows:

Definition 5.2 (Probabilistic Threshold Query (PTQ))
Let DB be a set of uncertain data objects described by pfvs having d uncertain dimensions and let $t \in [0, 1]$ be a probability threshold. Given d query intervals $[a_i, b_i]$ with $1 \leq i \leq d$ and $a_i, b_i \in \mathbb{R}$, a probabilistic threshold query (PTQ) returns all objects $\vec{v} \in \mathcal{DB}$ for which the following condition holds:

$$Pr(v_i \in [a_i, b_i], \forall i : 1 \leq i \leq d) \geq t.$$

Let us note that if we cannot specify a query interval for one of the attributes, we may assume that the attribute is allowed to have any value. In this case, the probability for this dimension is 1 which is the integral over the complete value set of the Gaussian density function. To compute a PTQ, the straightforward approach is to retrieve each pfv in the database DB and calculate the probability that the corresponding object has attribute values which are contained in the query area. If this probability is larger than t the object is part of the result set.

Formulating a PTQ often proves to be more complicated then necessary. Though the given query interval might be available, finding a useful threshold probability is often difficult. Thus, a PTQ might have to be repeated with varying threshold values until a reasonable result set is found.

To avoid this problem, we introduce a new type of uncertainty queries called probabilistic ranking queries (PRQs). A PRQ retrieves the k most likely data objects that might be placed in the given query interval. Specifying the number of results is usually much more intuitive and can easily be done by any user. In the ship example, a possible PRQ would be : "Retrieve the 10 ships which are most likely in the given area". Formally, a PRQ is defined as follows:

Definition 5.3 (Probabilistic Ranking Query (PRQ))
Let DB be a database of uncertain objects described by pfvs and let $k \in \mathbb{N}$ be a natural number. Given d query intervals $[a_i, b_i]$ with $1 \leq i \leq d, a_i < b_i, a_i, b_i \in \mathbb{R}$, a probabilistic ranking query (PRQ) over DB returns the smallest set of data objects $kSet(\vec{a}, \vec{b})$, having at least k elements, for which the following condition holds:

$$\begin{gathered} \forall p \in kSet(\vec{a}, \vec{b}), \forall q \in \mathcal{DB} \setminus kSet(\vec{a}, \vec{b}) : \\ Pr(p_i \in [a_i, b_i], \forall i : 1 \leq i \leq d) > \\ Pr(q_i \in [a_i, b_i], \forall i : 1 \leq i \leq d). \end{gathered}$$

If the number of result objects is not clear, PRQs can be extended to incremental PRQs which always retrieve the object having the next largest probability. Since the introduced query algorithms yields a close similarity to the query algorithm for nearest neighbor search described in [HS95], an extension to incremental queries is straight forward.

5.4 Processing Spatial Probabilistic Queries

In the previous section, we have introduced the Gaussian uncertainty model and queries on top of a set of uncertain data objects. We are now going describe algorithms for efficiently answering PRQs and PTQs on the Gauss-tree.

For query processing, we need a conservative approximation of the probability that any possible Gaussian which is stored in a node or in a certain subtree, can achieve over the given query area. As a formula, the conservative approximation of this probability $\hat{P}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(a,b)$ is given as:

$$\hat{P}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(a,b) = \max_{\mu \in [\check{\mu},\hat{\mu}], \sigma \in [\check{\sigma},\hat{\sigma}]} \{P_{\mu,\sigma}(a,b)\}$$

For efficient query processing, a closed formula for $\hat{P}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(a,b)$ without an explicit maximization process over two continuous variables is needed. To derive this closed form, we first of all derive the following lemma.

Lemma 5.1 *Let $[a,b]$ with $a < b$ and $a, b \in \mathbb{R}$ be a given query interval and let $\sigma \in]0, \infty[$ be a given standard deviation. Then, the Gaussian for the given σ having the maximum probability over the interval $[a,b]$ has the mean value: $\mu_{max} = \frac{a+b}{2}$.*
Furthermore, the probability of the Gaussian decreases monotonically with the distance of μ from μ_{max}.

Proof. *We can differentiate $P_{\mu,\sigma}(a,b)$ by μ and see that there is only one extremum μ_{max}. Furthermore, the limes of $P_{\mu,\sigma}(a,b)$ for $\mu \rightarrow \pm\infty$ is 0. Since $P_{\mu_{max},\sigma}(a,b) > 0$, $P_{\mu,\sigma}(a,b)$ is monotonic on both sides of the maximum.* $\square$

Based on that lemma we can state that the mean value $\mu^* \in [\check{\mu},\hat{\mu}]$ of the wanted conservative approximation is always the one closest to the middle of the query interval:

$$\mu^* = \max\{\check{\mu}, \min\{1/2(a+b), \hat{\mu}\}\}$$

To find the corresponding σ^* for the conservative approximation, we formulate the following lemma:

Lemma 5.2 *Let $[a,b]$ with $a < b$ and $a,b \in \mathbb{R}$ be a given query interval, let μ be a given mean value and let $[\check{\sigma}, \hat{\sigma}]$ be the interval of valid σ values with $0 < \check{\sigma} < \hat{\sigma}$. Then, we can maximize $P_{\mu,\sigma}(a,b)$ by selecting σ^* from $[\check{\sigma}, \hat{\sigma}]$ as follows:*

Case I *$a < b < \mu$:*

$$\sigma_{max} = -\frac{\sqrt{2\ln\left(\frac{\mu-b}{\mu-a}\right)(a-b)(2\mu-a-b)}}{2\ln\left(\frac{\mu-b}{\mu-a}\right)}$$

and $\sigma^* = \max\{\check{\sigma}, \min\{\sigma_{max}, \hat{\sigma}\}\}$.

Case II *$a \leq \mu \leq b : \sigma^* = \check{\sigma}$.*

Case III *$\mu < a < b$:*

$$\sigma_{max} = \frac{\sqrt{2\ln\left(\frac{\mu-b}{\mu-a}\right)(a-b)(2\mu-a-b)}}{2\ln\left(\frac{\mu-b}{\mu-a}\right)}$$

and $\sigma^* = \min\{\hat{\sigma}, \max\{\sigma_{max}, \check{\sigma}\}\}$.

Proof. ***Case I*** *We can differentiate $P_{\mu,\sigma}(a,b)$ for σ and receive the above formula for σ_{max} which is the only extremum in $]0,\infty[$. Examining the limes $\sigma \to 0$ and $\sigma \to \infty$, we observe that $P_{\mu,\sigma}(a,b)$ converges against 0 in both cases. Since $P_{\mu,\sigma_{max}}(a,b) > 0$, $P_{\mu,\sigma}(a,b)$ decreases monotonic on both sides of σ_{max}. Thus, σ^* can be chosen to be the closest value to σ_{max} in $[\check{\sigma}, \hat{\sigma}]$.*

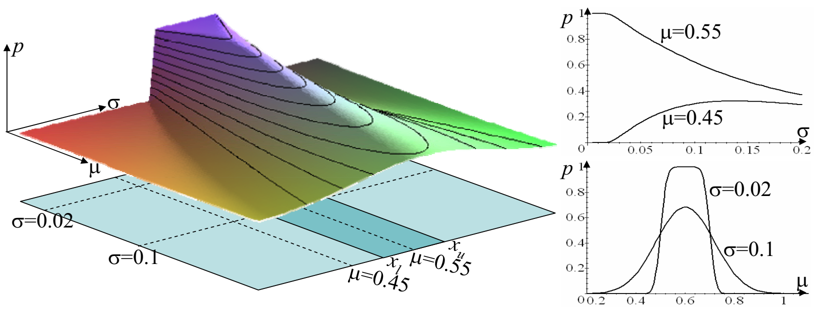


Figure 5.2: Visualization of probabilities for $\hat{P}_{\mu,\sigma}(a,b)$ in the μ-σ space.

Case II *In this case, μ is inside $[a,b]$ and if $\sigma \rightarrow 0$ then $P_{\mu,\sigma}(a,b) \rightarrow 1$. Since if $\sigma \rightarrow \infty$ then $P_{\mu,\sigma}(a,b) \rightarrow 0$ and there is no defined extremum, $P_{\mu,\sigma}(a,b)$ is monotonic and the smallest $\sigma \in [\breve{\sigma},\hat{\sigma}]$ causes the largest value for $P_{\mu,\sigma}(a,b)$.*
Case III *This case is symmetric to* ***case I****.* □

Using both lemmas, we can calculate $P_{\mu^*,\sigma^*}(a,b)$ which is the largest possible probability for any Gaussian stored in a given node or subtree of the Gauss-tree. Let us note that this bound is tight which means that there could be indeed a Gaussian in the node having exactly the calculated probability. Figure 5.2 displays the probabilities for a given query interval $[a,b]$ for arbitrary μ and σ.

5.5 Efficient Query Processing using the Gauss-Tree

After deriving a conservative approximation of the maximum probability of Gauss-tree nodes, we are now going to describe algorithms for query processing which are suitable for answering PTQs and PRQs in efficient time.

PTQs. The algorithm for answering PTQs traverses the Gauss-tree from the root node in a depth-first order. Thus, the algorithm starts with inserting the subtrees of the root node into a stack. Now, the algorithms always takes the first object from the stack until the stack is empty. If the object is a node the algorithm determines μ^* and σ^* and calculates $\hat{P}_{\check{\mu},\hat{\mu},\check{\sigma},\hat{\sigma}}(a,b)$ for each dimension. After multiplying the probabilities for each dimension the resulting approximation is compared to the threshold t. If the approximation is smaller than t, we can prune the corresponding subtree. If not, we must push the son objects of the node onto the stack. If the object on top of the stack is a pfv, we determine its probability for lying within the query area. If this probability is larger than t we have found a result and store it for output. Let us note that this algorithm is given for demonstrating that the Gauss-tree is applicable to PTQs as well. However, the main focus of this chapter are PRQs which are described in the following.

PRQs. For the answering PRQs, we employ the same idea as proposed in [HS95]. Instead of using a stack, the algorithm ranks the yet unprocessed entries of the Gauss-tree with a priority queue, which we will call entry queue. The entry queue has to be ordered in descending order w.r.t. to the largest probability value. Furthermore, we need a second priority queue to store the k best results being retrieved so far. This second queue is ordered in ascending order which means the result pfv having the smallest probability is always on top of the queue. We will refer to this queue as result queue.

Figure 5.3 denotes the algorithm in pseudocode. The algorithms starts with pushing the root node onto the entry queue with a probability of 1. Afterwards, we always remove the top object from the entry queue until the entry queue is empty or the algorithms can be guaranteed to have found all valid results. If the top element is a inner node, we load all son nodes, calculate their conservative approximation probabilities and insert them into the entry queue w.r.t. to these probabilities. In the case, a leaf node is placed on top of the entry queue, the exact probabilities for all pfvs stored in the node are calculated and the objects are pushed on the entry queue as well. If the top element of the entry queue is a pfv, we check if the result queue

```
ProbabilisticRankingQuery(Query $q$, integer $k$)
  $entryQueue$: ascending priority queue;
  $resultQueue$: descending priority queue;
  $entryQueue.insert(root, 1)$;
  WHILE $notentryQueue.isEmpty()$ or
    $entryQueue.getFirst() > resultQueue.getFirst()$ DO
    $currentNode = entryQueue.removeFirst()$;
    IF $currentNode$ is a data node THEN
      FOR EACH $d$ in $currentNode$ DO
        $prob =$ calculate probability of $d$ w.r.t. $q$;
        IF $resultQueue.size() < k$ THEN
          $resultQueue.insert(d, prob)$;
        ELSE IF $resultQqueue.getFirst() < prob$ THEN
          $resultQueue.removeFirst()$;
          $resultQueue.insert(d, prob)$;
        END IF
      END FOR
    ELSE IF $currentNode$ is a directory node THEN
      FOR EACH entry $e$ in $currentNode$ DO
        $prob =$ calculate probability of $e$ w.r.t. $q$;
        $entryQueue.insert(e, prob)$;
      END FOR
    END IF
  END WHILE
  RETURN $result$;
```

Figure 5.3: Pseudocode of probabilistic ranking query.

threshold	Sequ. Scan	x-b. tree	GX-tree	Gauss-tree
0.5	200.3	140.0	139.0	137.6
0.75	258.1	101.4	101.6	101.1

Table 5.1: Comparison of average query time on DS1 for PTQs.

already contains k results. If not, we can add the pfv as a possible result. If we have already encountered k pfvs, we must check if the new pfv has a larger probability than the top element of the result queue. If the new pfv is a more likely result than the top of the result queue, the top of the result queue is removed and the new pfv is added to the result queue. The algorithm can be terminated if the top of the result queue has a larger probability than the top of the entry queue. In this case, it can be guaranteed that there are no pfvs which have a larger probability than the k objects in the result queue. Let us note that this algorithm is optimal since it guarantees that no unnecessary nodes are read from the hard drive.

5.6 Experimental Evaluation

In our experimental evaluation, we implemented the Gauss-tree and its comparison partners in Java 1.5. To make the results reproducible, we measured the CPU time and counted logical page accesses on the hard drive. For calculating the complete query time, we assumed a hard drive having 6 ms access time and 50 MB/s transfer rate.

We employed two datasets. The first dataset (DS1) was a set of 100.000 1-dimensional Gaussians for which the μ and σ values were randomly generated. Dataset 2 (DS2) was taken from the TIGER[1] database containing 2D spatial coordinates of landmarks in the US. For DS2, we used the county of Sacramento having 62.182 objects. Since we did not have any uncertainty

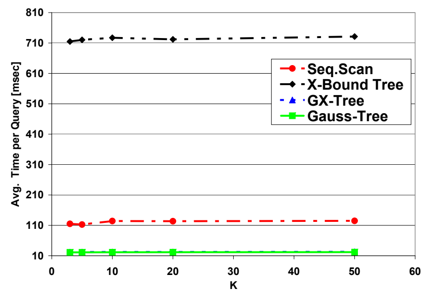
[1]available at http://www.census.gov/geo/www/tiger

values, we generated random standard deviations for each of the coordinates. We randomly generated 200 query intervals for DS1 or 200 query rectangles for DS2.

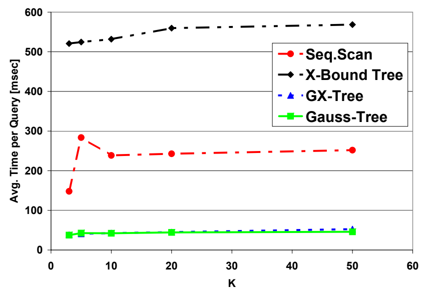
To have a baseline comparison partner, we compared our methods to a sequential scan over the complete database. Additionally, we implemented the method for symmetric and smooth variance-monotonic distribution functions being described in [CXP+04]. We will refer to this method as "x-bounds-tree". To extend the x-bound-tree to the multi-dimensional case, we pruned each dimension separately, i.e. when testing the pruning criteria, we assumed a maximum probability of 1 in all other dimensions. Let us note that this is not optimal, since multiplying several dimensions usually decreases the probability. However, since the method does not allow to derive a concrete maximum probability for any dimension but only checks if the closest bound is violated, this method is a feasible solution. For demonstrating the effect of our splitting and insertion method, we implemented a Gauss-tree employing the split and insertion algorithm of the x-tree [BKK96] to which we refer to as GX-tree.

Our first experiment compares the average query time for 200 PTQs on DS1. Table 5.1 compares the average elapsed time for a PTQs with $t = 0.5$ and $t = 0.75$ for all 4 methods. The results indicate that all indexing techniques were capable to answer the given queries significantly faster than the sequential scan. However, all three index structures used almost exactly the same number of accessed pages for each query and used very similar CPU times. Therefore, we can conclude that the more exact approximations of the Gauss-tree do not yield an advantage when answering PTQs and the x-bounds are an efficient method for this type of queries.

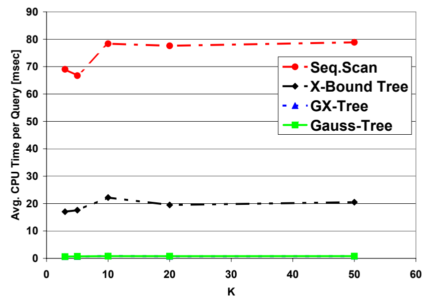
The main part of our experiments was examining the performance of the Gauss-tree when answering PRQs. To process PRQs on the x-bounds-tree, we had to find a way to rank pages w.r.t. this maximum probability. This



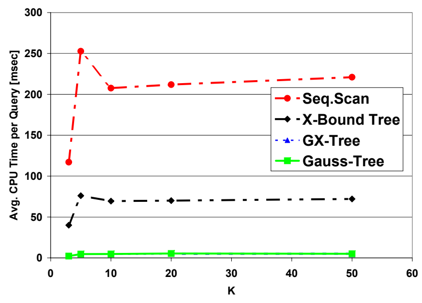
(a) Average query time on DS1.



(b) Average query time on DS2.



(c) Average CPU time on DS1.

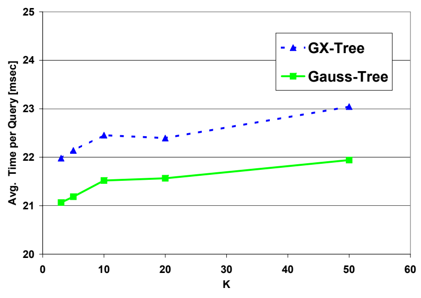


(d) Average CPU time on DS2.

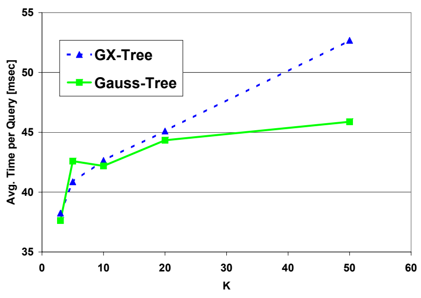
Figure 5.4: Complete runtime (above row) and CPU time (lower row) for PRQs for varying values of k.

is a problem because the described method only determines if a page can contain a pfv having a larger probability than some threshold. In order to apply ranking, we had to find a way to determine the largest probability any object in a node could have in the query interval. We solved this problem by searching the proposed ratio table for the closest x-bound to the query interval which is still outside the interval. The x corresponding to this bound was used to rank the entry queue. Let us note that the decision about pruning a node was done as proposed for PTQs in [CXP+04].

In our first experiment for PRQs, we tested all four methods for varying values of k on both datasets. The results are displayed in figure 5.4. The upper row of figure 5.4 displays the average elapsed time per query, i.e. CPU time together with calculated IO costs, and the lower row displays the



(a) Average query time DS1.



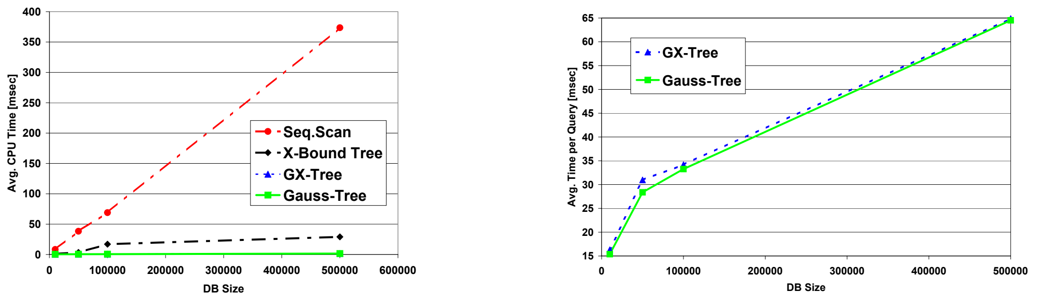
(b) Average query time on DS2.

Figure 5.5: Average time for a PRQs for the Gauss-tree and the GX-tree.

observed CPU time only. As a result it can be observed that the Gauss-tree and the GX-tree retrieved the query results between 8 to 10 times faster than the sequential scan. Our adapted version of the x-bound tree worked even worse than the sequential scan w.r.t to the all over query time. However, the x-bound-tree clearly beats the sequential scan w.r.t. CPU time. Finally, the better selectivity of the Gauss-tree related methods achieves an average CPU time which again is orders of magnitudes smaller than the comparison partner on both datasets.

Due to the overwhelming speed up compared to the sequential scan, the figure cannot display the difference between the Gauss-tree and the GX-tree. To still demonstrate that our new split heuristic was capable to improve the structure of the tree, we display figure 5.5 which is a zoomed version of figure 5.4(a) and 5.4(b). As it can be seen the new split heuristic additional decreased the average complete query time by an additional msec.

To demonstrate that our method scales well even for larger datasets, we posed PRQs with $k = 3$ on dataset DS1 and increased the size of the database from 10.000 to 500.000. The results are displayed in figure 5.6. Again our adaption of the x-bound tree for PRQs did not function very well. However, the Gauss-tree and the GX-tree again display a considerable speed up which is growing with the size of the database as shown by experimental results



(a) Average query time on DS1. **(b)** Zoomed average query time on DS1.

Figure 5.6: Average time for a PRQs on DS1 with increasing database size.

illustrated in 5.6(a). In order demonstrate that our new split heuristic was capable to improve the structure of the tree, we display figure 5.6(b) which is a zoomed version of figure 5.6(a). Thus, we can conclude that the Gauss-tree is especially well suited for very large datasets of uncertain objects modeled by Gaussians.

To conclude, the performance of the Gauss-tree for answering PTQs was rather similar to the x-bound tree in its original use. However, when answering PRQs the Gauss-tree outperformed all comparison partners by orders of magnitude. Furthermore, our novel split heuristic introduced in Chapter 3 further improved the structure of the tree when answering PRQs.

5.7 Conclusions

In this chapter, we introduced the Gaussian uncertainty model for describing uncertain spatial data objects. This model describes a spatially uncertain data object as probabilistic feature vector (pfv) consisting of a mean value and a standard deviation for any uncertain feature value. Assuming a Gaussian density distribution based on these parameters, we can now determine the probability for any data object for being contained in a certain interval or (hyper-) rectangle. Applications for spatially uncertain objects are sensor network and moving object database systems where the exact feature value cannot be constantly monitored. To query databases of uncertain objects, we can pose probabilistic queries like probabilistic threshold queries (PTQs). A PTQ retrieves all data objects in a database that are contained in the query rectangle with a larger probability than some probability threshold t. Since the threshold is often difficult to decide, we introduced probabilistic ranking queries (PRQs) which retrieve the k data objects in a database that are contained in the query rectangle with the highest probability. To answer both types of queries in efficient time, we derived the conservative approximation of probability for a node w.r.t. a given query range. This tight approximation is the basis of the described algorithms for answering PTQs and PRQs. In our experimental evaluation, we compared the Gauss-tree on both types of queries to 3 comparison partners on one artificial and two real-world datasets with artificial uncertainty. The results demonstrates that the Gauss-tree achieves a query performance which is comparable to existing methods on PTQ. For the new query type of PRQs the Gauss-tree clearly outperforms established methods which were modified to answer PRQs.

Chapter 6

Effective Similarity Search in Multimedia Databases using Multiple Representations

Similarity search in large multimedia databases is an important issue in the modern multimedia environment. Multimedia objects such as music videos usually consist of multiple representations like audio or video features. Since each representation may be of significantly different quality for a given multimedia object, similarity search methods could greatly benefit from taking these multiple representations into account. Therefore, in this chapter we develop an intelligent similarity search technique that considers all available representations of the database objects, and is capable of judging the importance of a representation automatically depending on a given query object. This chapter starts in Section 6.1 with an introduction into the multi-represented similarity search in multimedia databases. We review related work in Section 6.2. Details of our novel method are presented in Section 6.3. In Section 6.4, we present an experimental evaluation. We conclude this chapter in Section 6.5 with a short summary.

6.1 Introduction

With the rapid development of digital technologies, computer networks and Internet, the amount of multimedia data is growing enormously because digital videos are easily copied and distributed. Efficient and effective similarity search in such huge amounts of multimedia data has become a major issue in several important applications such as video copyright matters and multimedia retrieval [TKR99]. In fact, video similarity detection has been proposed as a promising approach for copyright issues which is complementary to the approach of digital watermarking [HB01]. In addition, video similarity search is the key step towards content-based video retrieval. As a consequence, lots of work has been done in the field of similarity search in multimedia databases so far (e.g. [CZ02b], [IL00], [NWH01b], [TKR99]).

Multimedia data such as movies can usually be viewed as multi-represented objects, i.e. for each object there are multiple representations modeling different features of the object. For example, for music videos, we can collect audio features, such as pitch [TK00] or rhythm [TC02], and video features, such as color histograms or textures [AY99]. Each of these multiple representations models a different aspects of a music video. Obviously, the effectiveness of similarity search methods could greatly benefit from taking multiple representations into account. However, most existing approaches for multimedia similarity search do not consider the multi-represented structure of multimedia objects but usually use one representation for similarity search only.

In this chapter, we propose a novel framework for video similarity search that takes the multi-represented nature of the data objects into account. In particular, our framework is able to integrate multiple representations such as audio and video features into the query processing. The most important issue for multi-represented similarity search is the weighting of each representation, i.e. the decision "how significant is a given representation for a given query object". We propose methods for this task that can be applied to both types of summarization techniques, i.e. higher-order and first-order summarization, that are commonly used in multimedia similarity search (cf. Section 2.1.5 in Chapter 2 for details). In addition, we propose a method for combining multiple representations for similarity search by weighting each representation. A broad experimental evaluation of our methods using a database of music videos demonstrates the benefit of our methods for similarity search in multimedia databases.

6.2 Related Work

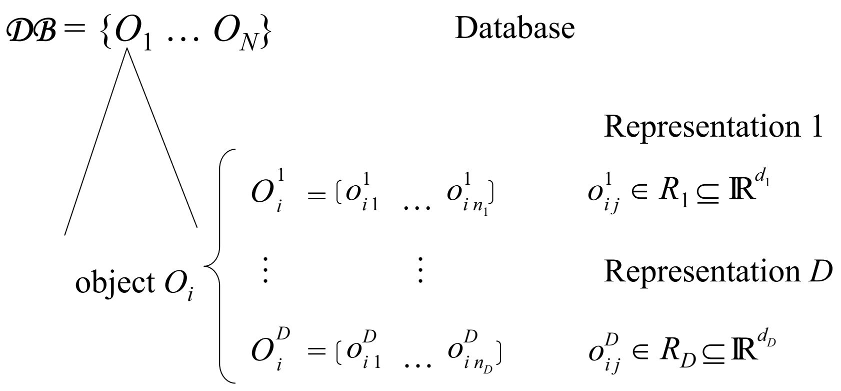
Similarity Search Based on Multiple Representations. Recently, some work has been done on multi-represented similarity search in multimedia databases as discussed in Section 2.1.3. The interactive search fusion method [SJL+03b] provides a set of fusion functions that can be used for combining different representation. Let us note that this technique is supervised, i.e. requires strong interaction with the user, which is not always desirable since it requires the user to understand the basic concepts of the method. Moreover, the proposed technique does not support individual weighting for each query object.

In [NWH01b], a template matching method based on the time warping distance is presented. This approach can measure the temporal edit similarity. However, temporal order is not necessary in many applications. In addition, this technique is not applicable to large databases because it is linear in the number of feature vectors of all video and audio sequences in the database. The authors of [BKS+04] proposed two methods for improving the effectiveness in a retrieval system that operates on multiple representation of 3D objects. These techniques need a set of labeled data in order to measure entropy impurity. Such set of labeled data is not always available. Furthermore, these techniques are not directly applicable to set-valued objects like videos.

Summarization Techniques. Usually, multimedia objects like video clips or pieces of audio consists of thousands or even millions of feature vectors. In order to handle such data efficiently, summarization techniques are usually applied to the original data, i.e. the original feature vectors are grouped together and each group is represented by a *summarization vector* or *summarization representative*. Then similarity is defined based on these summarizations. In general, we can distinguish two classes of summarization techniques: higher-order and first-order summarization. A detailed introduction into summarization techniques was given in Section 2.1.5 of Chapter 2. Our weighting approach can use an arbitrary summarization technique. Furthermore, it is rather general because it does not depend on a particular summarization technique.

6.3 Multi-Represented Similarity Search in Multimedia Databases

In the following, we assume $\mathcal{DB}$ to be a database of N multimedia objects. Each object $O_i \in \mathcal{DB}$, $i = 1, \ldots, N$, is represented by a given set of D representations $R_1, \ldots, R_D$, where each representation is a feature space, i.e. $R_i \subseteq \mathbb{R}^{d_i}$, and $d_i \in \mathbb{N}$ denotes the dimensionality of the feature space of representation R_i $(1 \leq i \leq D)$. The j-th representation of O_i is denoted by O_i^j, i.e. $O_i = (O_i^1, \ldots, O_i^D)$. We further assume that each representation O_i^j of O_i consists of a series of feature vectors of length n_j, i.e. $O_i^j = (o_{i\,1}^j, \ldots, o_{i\,n_j}^j)$ with $o_{i\,l}^j \in R_i$. The definitions are summarized in Figure 6.1. In addition, we assume that the distances within each representation are normalized sufficiently over all representations, e.g. using any of the methods of [SJL+03b].



$\mathcal{DB} = \{O_1 \dots O_N\}$
Database
object O_i
Representation 1
$O_i^1 = [o_{i1}^1 \dots o_{i n_1}^1]$
$o_{ij}^1 \in R_1 \subseteq \mathbb{R}^{d_1}$
Representation D
$O_i^D = [o_{i1}^D \dots o_{i n_D}^D]$
$o_{ij}^D \in R_D \subseteq \mathbb{R}^{d_D}$

Figure 6.1: Basic notations.

In order to combine multiple representations within the similarity evaluation, we have to determine for each object $O_i \in \mathcal{DB}$ and for each of its representations O_i^j a weight for each of the n_j feature vectors $o_{i\,1}^j, \dots, o_{i\,n_j}^j$.

Having weights for each feature vector of each representation of each object, we can use any common distance measure between sets of points such as the Hausdorff distance in order to compute a weighted distance between two multi-represented multimedia objects. We will first introduce novel methods to determine the weights for a feature vector of a given representation and then describe how these weights can be used to improve similarity search on multimedia objects.

6.3.1 Weighting Functions for Summarizations

As described above, a multimedia object usually consists of a large set of feature vectors per representation. For efficiency reasons, these large sets of feature vectors are usually summarized within each representation. The derived summarizations can be classified as first-order or higher-order summarizations (cf. Chapter 2, Section 2.1.5). Thus, the feature vectors $o_{i\,1}^j, \dots, o_{i\,n_i}^j$ of object $O_i \in \mathcal{DB}$ of representation R_j are representative points of the derived summarizations $S_{i\,1}^j, \dots, S_{i\,n_i}^j$. In the following, an original point p

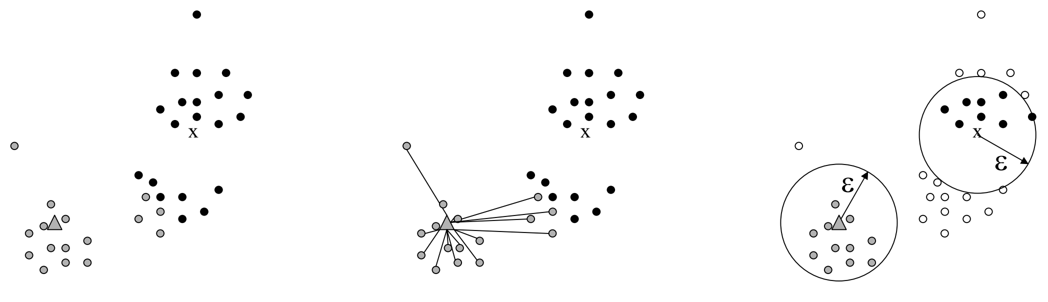
belongs to a summarization S if it is a member of the according cluster (in case of higher-order summarizations) or if the according representative of S is the representative with the lowest distance to p among the representatives of all summarizations.

Since different users may have a different notion of similarity among videos, it is desirable to consider this diversity in a best possible way when defining a similarity measure between multimedia objects. For our multi-represented approach, we have to take this diversity into account when we design a weighting function for the feature vectors of each representation. Thus, in the following, we present four methods to determine weights for representative feature vectors of a summarization that rates the significance of these summarization vectors in order to represent the according original feature vectors. The different weighting functions reflect different notions of similarity. Note that the weighting factor of each representative point is evaluated for each data object and each representation separately.

A Weighting Function Based on Support. The idea behind our first weighting function is that each summarization vector represents a given amount of original feature vectors. This amount is a good indication on the significance of this representative, i.e. how good this summarization represents the original feature vectors. Thus, in our first approach, the weight of the l-th feature vector $o^j_{i\,l}$ of the j-th representation of object $O_i \in \mathcal{DB}$, denoted by $\mathcal{W}_{\text{supp}}(o^j_{i\,l})$, is computed by the fraction of points that are represented by $o^j_{i\,l}$. Formally, if $|S^j_{i\,l}|$ denotes the fraction of original points that are summarized by $S^j_{i\,l}$, then the weight of the representative $o^j_{i\,l}$ is computed by

$$\mathcal{W}_{\text{supp}}(o^j_{i\,l}) = |S^j_{i\,l}|/n_j.$$

This weighting function is illustrated in Figure 6.2(a). The original points that contribute to the weight of the representative denoted by "△" are shaded in light gray, whereas the original points that contribute to the weight of



(a) $\mathcal{W}_{\text{supp}}$ **(b)** $\mathcal{W}_{\text{qual}}$ **(c)** $\mathcal{W}_{\text{local}}$

▲ x representatives	○ objects that are taken into account for weighting of ▲	● objects that are taken into account for weighting of x

Figure 6.2: Illustration of three different weighting functions.

the representative denoted by “x” are shaded in black. The weight for the representative denoted by “△” is simply computed by the fraction of gray points. The weight for the representative denoted by “x” is computed by the fraction of black points.

A Weighting Function Based on Specific Quality Measures. The first weighting function only considers the number of objects the given summarization vector represents. However, it does not take the distances to the representative object into account. For example, consider a representative point r_l representing l objects rather bad, i.e. the average distance of the l points to their representative r_l is significantly high, and a representative point r_k representing $k < l$ points significantly better, i.e. the average distance of the k points to their representative r_k is significantly low. Using our first weight function, r_l would be weighted higher than r_k (since $k < l$) although this contradicts the intuitive aim of our weighting function. A better idea might be to consider the distances of the original points within one summarization to their representative.

Usually, the summarization is generated optimizing a specific quality function. For example, for higher-order summarizations, the summarization is derived from a clustering algorithm such as k-means or EM, which optimizes a clustering quality criterion (e.g. TD^2, log-likelihood). In case of first-order summarization techniques, we can e.g. use the method described in [CZ02a] and the according quality function. Our second quality measure is based on the quality criterion upon which the summarization is generated. Intuitively, a summarization vector with high representative power should be weighted high.

Let $CQ(o^j_{i\,l})$ be the quality measure for the l-th summarization vector of the j-th representation of object $O_i \in \mathcal{DB}$, based on which the summarization is generated, e.g. TD^2 in case of higher-order features generated by k-means. Then, the weight of $o^j_{i\,l}$ is computed by:

$$\mathcal{W}_{\text{qual}}(o^j_{i\,l}) = CQ(o^j_{i\,l}).$$

An example of this weighting function is visualized in Figure 6.2(b). The weight for the representative denoted by "△" is computed by e.g. the average distance of the original objects in its summarization to the representative.

A Weighting Function Based on Local Neighborhood. The second weighting function takes each original object into account when computing the weights for the derived summarizations. However, the original multimedia objects may contain some noise points, e.g. feature vectors that do not fit properly to any summarization, or — in case of an ineffective summarization procedure — one summarization may contain feature vectors of different clusters. In general, due to noisy original objects, the second weighting strategy may also fail. In this case, it would be more reliable to rate the weight of a representative point r based only on the original points in the local neighborhood of r. Our third weighting function follows this idea.

Let $\mathcal{N}_\varepsilon(r^j_i) = \{q^j_i | dist(r^j_i, q^j_i) \leq \varepsilon\}$ be the ε-neighborhood of a representative r^j_i of the i-th database object $O_i \in \mathcal{DB}$ in the j-th representation R_j. Let us note that $\mathcal{N}_\varepsilon(r^j_i)$ only contains original feature vectors q^j_i of O_i in representation R_j. We define the weight of $o^j_{i\,l}$ by the number of objects in its local neighborhood, formally

$$\mathcal{W}_{\text{local}}(o^j_{i\,l}) = |\mathcal{N}_\varepsilon(o^j_{i\,l})|/n_j.$$

This weighting function is illustrated in Figure 6.2(c). The original points that contribute to the weight of the representative denoted by "△" are again shaded in light gray, whereas the original points that contribute to the weight of the representative denoted by "x" are shaded in black. Original points that do not contribute to the weight of any summarization vector are shaded in white. The weights for both representatives are derived by the number of original points within their ε-neighborhood, normalized by n_j.

A Weighting Function Based on Entropy. The three weighting functions which we have introduced so far are rather local in the following sense: in order to compute the weight of a representative o of a summarization S_o, they only consider the objects that are summarized by S_o, i.e. belong to S_o. However, it may be more appropriate to consider all original features of a given representation R_i in order to rate a summarization vector o^i of this representation. Our fourth weighting strategy follows this idea.

When computing the weight of a summarization vector o^i of a representation R_i, we want to take the distances of all original feature vectors $q^i_1, \ldots, q^i_m$ of representation R_i to o^i into account. In fact, the distances of q^i_l to o^i can be considered as a random variable x following a Gaussian distribution $G(x)$. The information content of such a random variable can be measured by its entropy. For example, if the entropy of the variable x equals 1, the distances $dist(q^i_l, o^i)$ are randomly distributed, whereas if the entropy of the variable x is considerably low, the distances $dist(q^i_l, o^i)$ are most likely densely packed around the mean value of x and thus, o^i is a good representation of the vectors $q^i_1, \ldots, q^i_m$. Figure 6.3 illustrates two Gaussians with different standard deviations derived from two summarizations of different quality. The Gaussian displayed in the upper part of Figure 6.3 has a lower deviation because

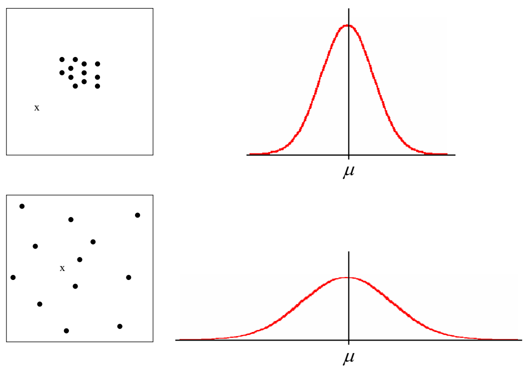


Figure 6.3: Different Gaussian distributions of distances from original objects to a summarization vector x.

the summarized original feature vectors are clustered. Its entropy will be considerably lower than the entropy of the Gaussians depicted in the lower part of Figure 6.3 which has a considerably higher standard deviation. This is due to the randomized distribution of the summarized feature vectors in the lower example.

Formally, let $x_{o^i} = \{dist(o^i, q_l^i) \mid 1 \leq l \leq m\}$ be a random variable. The Gaussian distribution $G(x_{o^i})$ of this random variable x_{o^i} is represented by the mean

$$\mu_{G(x_{o^i})} = \frac{\sum_{l=1}^{m} dist(o^i, q_l^i)}{m}$$

and the standard deviation

$$\sigma_{G(x_{o^i})} = \sqrt{\frac{1}{m} \cdot \sum_{j=1}^{m} (dist(o, q_j) - \mu_{G(x_{o^i})})^2}.$$

The entropy of x_{o^i} is then defined as

$$\mathcal{H}(x_{o^i}) = \int_{-\infty}^{+\infty} G(x_{o^i}) \cdot \log G(x_{o^i}) \; dx_{o^i}.$$

Let $o_{i\,l}^{j}$ be the l-th summarization vector of object $O_i \in \mathcal{DB}$ in representation R_j and let $x_{o_{i\,l}^{j}}$ be the random variable built by the distances of the original features of O_i in representation R_j to $o^j_{i\,l}$ as defined above. The weight of $o^j_{i\,l}$ is defined as the entropy of the random variable $x_{o^j_{i\,l}}$, formally

$$\mathcal{W}_{\text{entropy}}(o^j_{i\,l}) = 1 - \mathcal{H}(x_{o^j_{i\,l}}).$$

The weighting function evaluates to zero if the entropy equals 1, i.e. the distances are distributed randomly. On the other hand, the weighting is near 1 if the distance distribution has a small standard deviation, i.e. the original feature vectors are considerably dense around the summarization vector.

Let us note, that we can efficiently calculate the entropy by using an appropriate five-order polynomial approximation that depends on the mean and standard deviation.

6.3.2 Combining Multiple Representations for Similarity Detection

Having defined a weighting function for each summarization vector for each representation of a database object, we can combine multiple representations for the process of similarity detection. The key step for efficient similarity search is the design of a dedicated distance measure that takes the weights of each summarization vector into account.

In general, we can adapt any distance measure that has been designed for multimedia objects to consider the weights of each feature vector. Let $O = (O^1, \ldots, O^D) \in \mathcal{DB}$ be an arbitrary database object and let $Q = (Q^1, \ldots, Q^D)$ be the query object. Furthermore, let $dist^i$ be the distance function for comparing the i-th representation of O and Q, i.e. O^i and Q^i. Then, the distance between query object Q and a database object O can be computed by

$$dist(Q, O) = \sum_{i=1}^{D} \lambda^i \cdot dist^i(Q^i, O^i).$$

The most important part is to determine the weight λ^i of representation R_i. Obviously, λ^i should be derived from the weights of summarization vectors of the i-th representation of the query object Q, i.e. from $\mathcal{W}(q_1^i), \ldots, \mathcal{W}(q_{n_i}^i)$. The use of the weights of the query object Q only rather is more intuitive than using the weights of both Q and O because we want to ensure that we find database objects that are most similar to Q. Thus, the weights of Q are much more important than that of the database object O.

Regarding the distance function which should be used on the summarizations in each representation, we propose to distinguish between higher-order summarizations and first-order summarizations. Of course, we can combine representations of higher-order summarizations with representations of first-order representations.

Higher-order Summarizations. For higher-order summarizations, we use the Hausdorff distance which is an approved and frequently used distance measure in multimedia similarity search to compute the similarity between a database object $O^i = \{o_1^i, \ldots, o_n^i\}$ and a query object $Q = \{q_1^i, \ldots, q_n^i\}$ w.r.t. a given representation R_i. In general, an arbitrary distance function appropriate for multi-instance objects is applicable as similarity but the Hausdorff distance can be efficiently supported by an index structure like M-tree [CPZ97] because it satisfies metric properties.

In fact, the Hausdorff distance (cf. Definition 2.1.4 in Chapter 2) relies on the distance of two specific summarizations, one from Q^i, say q_h^i, and one from O^i, say o_h^i. In other words, there are two summarizations $q_h^i \in Q^i$ and $o_h^i \in O^i$, such that $H(Q^i, O^i) = dist(q_h^i, o_h^i)$. Then the weight of the i-th representation λ^i is determined by the the weight of q_h^i, formally

$$\lambda^i = \mathcal{W}(q_h^i).$$

Let us note that the distance function $dist(a, b)$ between two summarization representatives a and b can be arbitrary. If the summarization representatives are feature vectors, e.g. derived by k-means clustering, any common distance measure such as the Euclidean distance can be used. If the summarization technique generates Gaussian distributions, e.g. using EM clustering, we use the Kullback-Leibler distance [IL00].

First-order Summarizations. For first-order summarizations, we use the distance function proposed in [CZ02a] called ranked ViSig Similarity (VSS). This similarity measure relies on a set of distances between summarizations of the query Q and a database object O in each representation. Analogously to higher-order features, we weight each distance with the weight of the participating query summarization.

6.4 Experimental Evaluation

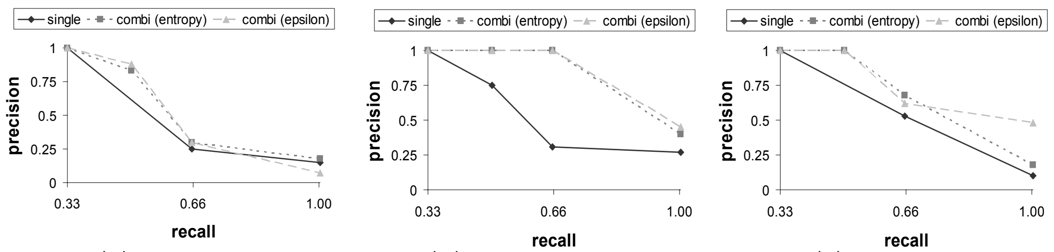
All experiments were performed on a workstation featuring a 1.8 GHz Opteron CPU and 8GB RAM. We evaluated our concepts using a database of 500 music videos recorded from various TV stations. The average length of a video clip within our collection is 4 minutes and 5 seconds. We extracted the image representations of the videos on a per-frame basis, i.e. we generated 25 features/second for PAL and 30 features/second for NTSC videos. From each image, we extracted four representations, namely a color histogram and three textural features. For the color histogram, we used the HSV color space which was divided into 32 subspaces, 8 ranges of hue and 4 ranges of saturation. The textural features were generated from 16 gray-scale conversions of the images. We computed contrast, entropy and inverse difference moment using the co-occurrence matrix [HSD73]. For extracting the audio features, we divided the audio signal of a video clip into short time frames, each having a length of 1/50 second. Every audio frame is represented by two features in the time- and frequency-domain. We computed autocorrelation and threshold-crossing for the time-domain, spectral flux and mel-frequency cepstral coefficients for the frequency-domain [TC02].

6.4.1 Multi-Represented vs. Uni-Represented Similarity Search

First, we show that multi-represented similarity search is usually more effective than similarity search using only one representation. In addition, we show in this subsection, that weighting the different representations yield a significant benefit compared to un-weighted multi-represented similarity detection.

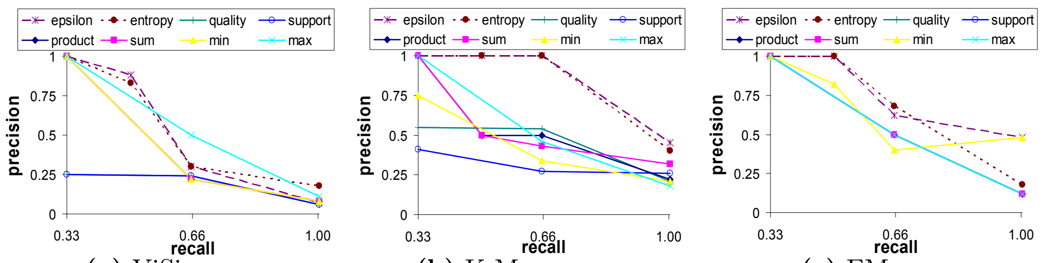
In a first experiment, we performed video similarity search. As setup step, we picked 50 query videos from our database and manually selected a set of videos which are similar to the query videos. We compared recall and precision achieved on the best single representation to the query result computed by using the ε-neighborhood and entropy weighting functions. Furthermore, we investigated the performance of our weighting strategies on three summarization techniques, namely video signatures (ViSig), K-Means and expectation maximization (EM). The results of this comparison is depicted in Figure 6.4. For all evaluated summarization techniques, we observed a significant performance improvement when using multiple representations in comparison to the best single representation. Furthermore, our weighted approach leads to better results on all considered summarization techniques.

Using the same test setup as described before, we compared different standard combination techniques for multi-represented objects to our weighted combination method is shown in Figure 6.5. We investigated the performance of commonly used standard combination techniques such as product, sum, minimum and maximum. In most cases, our weighted approach is more effective than the standard combination algorithms. Especially the ε-



(a) ViSig (b) K-Means (c) EM

Figure 6.4: Precision vs recall for different summarization techniques on best single representation and two best weighting functions.



(a) ViSig (b) K-Means (c) EM

Figure 6.5: Precision vs recall for different summarization techniques on standard combination strategies and proposed weighted combination strategies.

neighborhood and entropy weighting methods show good precision and recall values for all considered summarization strategies.

6.4.2 Multi-Represented Similarity Search Applications

In the following, we identify two common applications that may pose different challenges to multimedia similarity search techniques and propose the most appropriate weighting functions for these tasks.

Application 1: Finding Similar Videos. Our first application addresses copyright issues. In order to detect plagiarism, we want to find videos that are similar to a given query video. We argue that in this application, similarity should be considered more locally because several representations are usually almost identical. This is the case if e.g. the image or audio part of a video is encoded in different resolutions or sampling rates. To distinguish these videos from the rest of the database, it is necessary to examine a small neighborhood. Otherwise, we would obtain results which are similar, but do not violate the copyright.

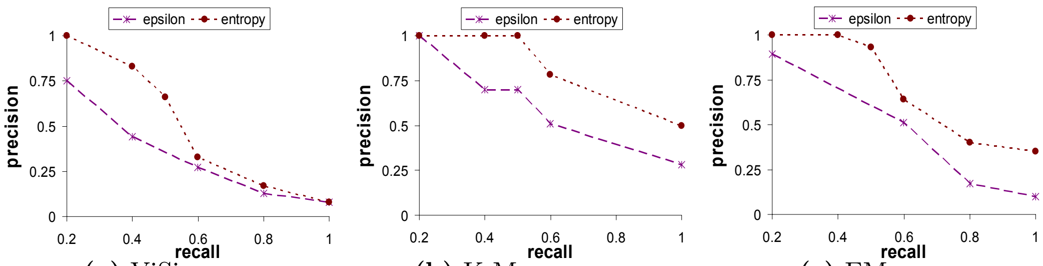
The ε-neighborhood weighting function follows this idea and can successfully be applied for this task as shown in Figure 6.4 and Figure 6.5.

Application 2: Finding Videos of a Given Artist. In our second application, we address content-based multimedia retrieval in music video databases. Given a query video of a specific artist, we want all videos of this artist in our database. Obviously, in this application, a more global notion of similarity is necessary.

In order to demonstrate this idea, we selected a set of 20 query videos associated with different artists. For each video in our query set, we extracted all videos of the same artist from our database. The results of our artist search are depicted in Figure 6.6. In all experiments, the entropy-based weighting function outperforms the ε-neighborhood approach. This can be explained by the fact that the entropy weighting function takes all distances into account in opposite to the local character of the ε-neighborhood function.

6.5 Conclusions

Similarity search in multimedia databases can be improved by using multiple representations of the multimedia objects. When searching for similar videos, one can e.g. use audio features such as rhythm and pitch as well as video features such as color histograms and textures.



(a) ViSig (b) K-Means (c) EM

Figure 6.6: Precision vs. recall for different weighting strategies when performing similarity search for videos of the same artist.

In this chapter, we presented a method for effective similarity search in multimedia databases that takes multiple representations of the database objects into account. In particular, we proposed several weighting functions for summarization vectors of different representations of each database object. Our concepts are independent of the underlying summarization method and compute a weight for each summarization vector of each representation for each object separately. Using these weighting factors, we further show how well-known distance measures for non-multi-represented, multi-instance objects can be adopted to multi-represented objects. In our experiments, we evaluated the proposed methods and showed the benefits of our approach.

Part III

Data Mining Techniques

Chapter 7

Using Uncertainty to Provide Privacy Preservation for Distributed Clustering

Privacy preservation is a new area in data mining research that deals with obtaining valid data mining results without learning the underlying data. In this chapter we introduce a novel method for clustering distributed data that achieves an arbitrary level of privacy preservation through the obfuscation of the original data using aggregation by the mixture of Gaussians. This chapter starts with an introduction into privacy preservation for distributed clustering in Section 7.1. In Section 7.2, we survey related work on distributed and parallel clustering. In Section 7.3, we describe our novel privacy-preserving clustering algorithm that describes original data by uncertain models. Section 7.4 provides an extensive experimental evaluation of the performance and the accuracy of the proposed approach. In Section 7.5, we summarize this chapter.

7.1 Introduction

As discussed in Chapter 1, advanced application have often to perform data mining task on distributed data under privacy preservation requirements. A good distributed data mining framework performs data mining operations based on the type and the availability of the distributed resources. As suggested in [PK03], a distributed data mining solution consists of the following steps. First, a data mining algorithm is locally applied to each of the k sites separately and independently. The results are k local sets of patterns called *local models*. Second, the local models are transferred to a central server. The central server combines the local models to generate a *global model*. Third, the global model may optionally be sent back to local sites.

The data mining technique we address in this chapter is clustering which aims at partitioning the data objects into distinct groups (clusters) while maximizing the intra-cluster similarity and minimizing the inter-cluster similarity. Many clustering algorithms for the centralized approach have been proposed so far using different clustering notions, e.g. distribution-(or model-)based, center-based, or density-based (cf. [HK06] for an overview). In general, all those methods are applicable for a distributed solution as far as they produce a local model in Step 1 of the distributed data mining process that is as compact as possible but provides as much information as needed for building a global model in Step 2. Unfortunately, many traditional clustering algorithms produce a clustering that cannot be easily described by a simple prototype. For example, density-based clustering [EKSX96] detects clusters of arbitrary shape. However, describing a cluster having a complex shape might become quite expensive possibly causing large transfer rates. Thus, a local model should describe each cluster by a "suitable" prototype. Obviously, such prototyping should also meet privacy constraints. We argue, that the expectation maximization (EM) clustering algorithm provides exactly such prototypes. EM describes the dataset by a set of Gaussian distributions consisting of the cluster center (mean) and covariance matrix. The latter describes the density of points around the center of the cluster. If certain constraints are met, privacy is preserved because the exact values of the data objects cannot be retrieved from the distribution.

We propose a novel distributed clustering algorithm called DMBC (Distributed Model-Based Clustering). The local models are acquired using EM clustering. Since the necessary number of clusters on each site might be strongly varying, DMBC automatically determines a suitable number of local clusters based on privacy and performance constraints. The constraints control the maximum transfer volume that is allowed from an individual site and assure that each local data object is described as good as possible and prohibit the transfer of clusters that could lead to a violation of privacy aspects. To combine the local clusters at the central server, the aggregation step of DMBC can employ two variants of parametrization to either derive a global clustering offering k clusters or an arbitrary set of clusters that are considerably different from each other. In both cases, DMBC derives a meaningful global mixture model of Gaussian in efficient time. Our broad experimental evaluation shows that DMBC is a scalable solution for clustering in a distributed environment that achieves comparative results compared to a centralized EM-based approach.

7.2 Related Work

In the following, we will review recent work on parallel and distributed clustering. Parallel clustering is related to the problem of distributed clustering because the data objects are also distributed over several clients where a local clustering is performed. The local clusterings are merged to produce the final model. However, parallel clustering methods can control the assignment of data objects to each site. Thus, the merge step is usually less complex and implying different problems than the merge step of distributed data mining approaches. However, several recent approaches for distributed data mining are adoptions of parallel clustering algorithms and do not consider privacy preservation issue.

Parallel versions of k-means, k-Harmonic-Means, EM [DM00, FZ00], and DBSCAN [XJK03] are all not applicable within distributed environments because all methods rely on a centralized view of the data during or before the clustering is computed.

In [SS00] a parallel algorithm is proposed for clustering web documents distributed randomly over several sites. Any clustering algorithm can be used to generate local clusters. The entire local clusters are sent back to the server rather than compact prototypes. Clusters are merged if they share a given number of documents which is determined by deriving maximum-sized itemsets from the documents. Obviously, since all local documents are transferred to the server, this approach does not consider any privacy issues.

In [JKP04] a distributed version of DBSCAN [EKSX96] is presented. The local clusters are represented by special objects that have the best representative power. This representative power is based on two quality measures that take the density-based clustering concepts into account. For each representative, a covering radius and a covering number is aggregated for the global merge step. The performance of the proposed method is heavily dependent on the number of representatives. If it is chosen too small, the accuracy significantly decreases. Otherwise, the runtime increases due to high transfer cost. In addition, since real data objects are sent to the global server, this approach does also not consider any privacy issues.

In [JK99] a single-link hierarchical clustering algorithm for vertically distributed data is proposed. However, our new approach DMBC is focused on horizontally distributed data.

7.3 Distributed Model-based Clustering

In the following, we will refer to the clustering generated by the centralized approach as *centralized clustering* and call the clusters that are part of the global clustering *centralized clusters.*

7.3.1 Problem Analysis

As discussed above, the centralized solution has several drawbacks which led to the distributed approach where the data is clustered locally at each site. Afterwards only the information about the local clusters are transfered to a central server. The server can now reconstruct the global clustering which should be as similar to the centralized clustering as possible, by combining the local clusters. For this recombination, we can distinguish the following cases:

Case 1: All objects of a global cluster are found on a single local site. In this case, the global cluster can be spotted easily at the local site and should be added to the global clustering on the central site.

Case 2: The objects of a global cluster are spread over several sites. In this case, we have to distinguish:

Case 2(a): All objects are rather well described by some local cluster. In this subcase, a good clustering algorithm should discover which local clusters belong to the same global cluster and merge them at the central site.

Case 2(b): Some of the objects of a global cluster are locally considered as noise or as members of local clusters that are built from object predominantly belonging to other global clusters. These objects contribute to the wrong global cluster or noise.

Case 2(c): A cluster is distributed over several sites and none of them contains enough data objects for deriving a local cluster. In this case, the objects of the global cluster are considered as noise or parts of other clusters at each client site. Only by combining the local objects the global cluster would become visible.

Because of the last two subcases, it is quite difficult, if not impossible, to develop an efficient distributed clustering algorithm that transmits local clusters and exactly rebuilds the centralized clustering for all cases. Since some of the centralized clusters may only be discovered if the noise objects of several sites are combined, the clustering algorithm would have to transmit all objects that are not well described by any local cluster as well as the clusters. However, transmitting all these objects has two major drawbacks. First, privacy preservation gets almost impossible by transmitting single data objects. Second, with large amounts of noise, it becomes necessary to transmit large amounts of data as well and thus, the advantages of distributed clustering might get lost. However, since there is no other solution for this dilemma, a good distributed clustering algorithm should at least offer the possibility to adjust to the users preferences on privacy, performance and the degree of how good the derived distributed clustering corresponds to the centralized clustering. In the following, we will refer to the degree of how good a distributed clustering corresponds to the centralized clustering as the *agreement* of both clusterings. Note, that a good distributed clustering algorithm cannot guarantee an agreement of 100% in all scenarios as discussed above.

In the following, we describe a method, called Distributed Model-based Clustering (DMBC) that is based on the EM clustering of local sites. Instead of transmitting the complete local dataset, we only transmit a number of local Gaussians and their weights to the central site. Since a Gaussian distribution is represented only by a mean vector and a covariance matrix, the amount of transferred information is much smaller. Therefore, the needed bandwidth is much smaller. Furthermore, the Gaussians derived by EM are always built according to all underlying data objects and drawing detailed conclusions about individual data objects is not possible in almost all settings. Since we will additionally control the remaining cases, the privacy of local data objects is preserved. Thus, our method avoids the problem of building a global clustering and still derives a mixture of Gaussian distributions achieving a high agreement with the centralized clustering.

The DMBC algorithm proceeds in 4 steps:

Step 1: The local data on each client site is clustered using EM. Thus, the data at each client site is now represented by a small but descriptive set of Gaussian distributions and a distribution of weights over these Gaussians. The number of clusters for the EM algorithm is optimized automatically with respect to the parametrization controlling the privacy level and the transfer volume.

Step 2: The local Gaussians and weights are transmitted to the central site.

Step 3: Similar local Gaussians are joined to find a compact global distribution. Thus, each cluster in a global EM clustering is only represented by a single Gaussian.

Step 4: The calculated global clustering can optionally be transmitted back to the client sites.

7.3.2 Computation of Local Models

To cluster the data at each client site, we employ the EM algorithm as described in Section 2.2.1. The most important aspect of this step is the question how to choose the parameter k, i.e. the number of Gaussians that is used to describe the local data distribution. Since the data distribution can strongly vary between each site, simply selecting a global value for k as the expected number of global clusters might be rather inappropriate.

Therefore, our algorithm automatically determines a particular value k_i for each site S_i. Let us note that k_i not only influences the transfer volume, but also the privacy and exactness of the clustering as well. The larger k_i, the higher is the probability that a Gaussian is strongly influenced by only a few datasets. In this case, the privacy can be seriously jeopardized because it might be easy to approximate the instances that are represented by this Gaussian. On the other hand, a large number of k_i usually increases the tendency that all local data objects are well represented by a local Gaussian. To conclude, a very high value for k_i will increase the agreement between our derived clustering and the global clustering, but it will also increase the transferred data volume.

To find a clustering containing an appropriate number of clusters, we first of all introduce the parameter k_{max} describing the maximum number of Gaussians for each local site. k_{max} limits the maximum transfer from a local site to the central site in step 2 and thus can be derived from the available bandwidth. To measure the degree that all data objects are well represented by the given clustering, we introduce the function $cover(C_{i,j})$.

Definition 7.1 (Cover)
Let $M = C_1, \ldots, C_k$ be a mixture of Gaussians describing the density distribution within $\mathcal{DB}$. Furthermore, let $t \in [0, \ldots, 1]$ be a probability threshold. Then, the cover *of the model M, denoted by $Cov(M)$, is defined as follows:*

$$Cov(M) = |\{\ \vec{x}\ | \vec{x} \in \mathcal{DB} \cap \exists\ C_i \in M\ \ : P(C_i|\vec{x}) \geq t\}|$$

Intuitively, the cover is the number of data objects that provide at least a probability of t for some Gaussian in the clustering. Let us note that $Cov(M)$ is related but different to the log-likelihood $E(M)$ that is optimized by the EM algorithm.

The pseudocode of the algorithm localEM to derive a local clustering is depicted in Figure 7.1. The algorithm chooses the smallest clustering M that

```
localEM(Database $\mathcal{DB}$, Integer $k_{max}$)
  maxcover = 0;
  bestClustering = $\emptyset$;
  for $k := 1$ to $k_{max}$ do
    $M :=$ EM($\mathcal{DB}$, $k$);
    if $cover(M) = |\mathcal{DB}|$ then
      return $M$;
    end if
    if $cover(M) >$ maxcover then
      maxcover = cover;
      bestClustering = M;
    end if
  end for
  return bestClustering;
```

Figure 7.1: Algorithm for local clustering.

achieves a maximum cover by successively increasing k and testing the cover of the resulting clustering.

At last, we have to control the level of privacy, we need to ensure. Thus, we have to measure how far it is possible to draw conclusions about individual datasets from the found clustering C. Therefore, we define the so-called privacy score (PScore):

Definition 7.2 (Privacy Score)
Let $C_i \in M$ be a cluster that is described by a d-dimensional Gaussian determined by the mean vector μ_i and a covariance matrix Σ_i. Then, the privacy-score*, denoted by $PScore(C_i)$, is defined as follows:*

$$PScore(C_i) = \sum_{j=1}^{d} \Sigma_{j,j}$$

The idea of the $PScore(C_i)$ is quite simple. Only if the variance in each dimension is very small, it is possible to draw conclusions about the underlying feature vectors. Let us note that the local cluster description of a cluster determined by the EM algorithm is always built using the complete dataset. As a result, it is impossible to derive detailed information about single feature vectors even for clusters having small weights if the variance values are large enough for at least a single dimension. Thus, we define a privacy threshold τ_p that is the lower limit for the $PScore(C_i)$ of a cluster C_i that is allowed to be transferred. If a cluster C_i has a smaller privacy-score, i.e. $PScore(C_i) < \tau_p$, we do not transmit the cluster because it would be possible to conclude that there is at least one feature vector stored on the local site that strongly resembles the transferred mean value.

To conclude, at each site we determine the smallest EM clustering providing a maximum cover and afterwards transfer all clusters that do not violate the predefined level of privacy.

7.3.3 Computation of the Global Model

The purpose of this step is to combine the locally derived clusters to a distributed clustering describing the complete data distribution in a best possible way. The difficulty in this step is to find out which of the clusters are likely to describe the same global cluster. To find out which of the given local clusters should be joined, first of all we need a measure that describes the likelihood of two local Gaussians C_1 and C_2 to model the same global cluster. Simply, using the distance between mean vectors is not applicable here because the significance of this distance strongly decreases with increasing variance values. Therefore, we define a new measure that considers the dependency between variance and mean value, called mutual support.

Definition 7.3 (Mutual Support)
Let C_1, C_2 be two Gaussian determined by a mean vector μ_i and a covariance matrix Σ_i. Then the mutual support *of C_1, C_2 is given by:*

$$MS(C_1, C_2) = \int_{-\infty}^{+\infty} N_{\mu_1,\Sigma_1}(\vec{x}) \cdot N_{\mu_2,\Sigma_2}(\vec{x}) d\vec{x}$$

The probability density of a point $\vec{x} \in \mathcal{DB}$ within a Gaussian density distribution $C = (\mu_C, \Sigma_C)$ is computed in the following way:

$$N_{\mu_C,\Sigma_C}(\vec{x}) = \frac{1}{\sqrt{(2\pi)^d|\Sigma_C|}} e^{-\frac{1}{2}(\vec{x}-\mu_C)^{\mathbf{T}}(\Sigma_C)^{-1}(\vec{x}-\mu_C)}.$$

Let us note that $\vec{x}$ is a d-dimensional feature vector and thus $MS(\vec{x})$ is defined using the integral over all d dimensions. The mutual support has several characteristics that makes it well suited for measuring the similarity between two Gaussians. The larger the variance values become, the less steep are the probability density functions of the Gaussians and the less important is the distance between the mean values. Comparing a low variance distribution with a high variance Gaussian, will display a small mutual support. In the comparably small range where a low-variance distribution displays strong density the high-variance distribution provides only moderate density and in the large area where the high-variance distribution displays still moderate density, the density of the low-variance distribution decreases the product very strongly. Thus, the mutual support of two Gaussians specified by very similar mean values but quite different covariance matrices is also rather small.

After finding a method to compare two local Gaussians, we now start to determine which of the local clusters should be merged. To determine a distributed clustering from a set of local clusters C with a number of k global clusters, we can now proceed as described in Figure 7.2.

Another alternative for deriving a joined distributed clustering is to specify a threshold parameter τ and join all clusters displaying a mutual support of at least τ. In this case, all pairs of clusters (C_i, C_j) are marked if $MS(C_i, C_j) \geq \tau$. Again we first of all, find the marked pairs of clusters that are connected by common clusters and afterwards merge all clusters in this connected set. Therefore, both approaches are independent of the order the clusters are merged. After determining which clusters have to be joined, we still need to derive a common Gaussian from a connected set of local

```
globalMerge(SetOfLocalClusters $C$, Integer $k$)
  for each pair $(C_i, C_j) \in C$ do
    compute $MS(C_i, C_j)$;
  end for
  sort the pairs w.r.t. descending mutual support;
  mark the first $|C| - k$ pairs of clusters;
  build the transitive closure over the pairs having some
    common clusters and unite them into a common
    global cluster;
```

Figure 7.2: Algorithm for global clustering.

clusters. Therefore, we derive a new mean μ_C for a set of Gaussian clusters $C = \{C_1, \ldots, C_m\}$ that are specified by μ_i and Σ_i in the following way.

$$\mu_C = \frac{\sum_{k=1}^{m}(w_{C_k} \cdot \lambda(C_k) \cdot \mu_k)}{\sum_{k=1}^{m}(w_{C_k} \cdot \lambda(C_k))}.$$

Here, $\lambda(C_k) = Cov(M_l)$ where cluster C_k has its origin on site l, i.e. $\lambda(C_k)$ denotes the cover of site l. The entries of the covariance matrix for the ith line and the jth column are calculated as following:

$$\Sigma_C^{i,j} =$$

$$\frac{\int_{-\infty}^{+\infty}(\sum_{k=1}^{m} w_{C_k}\lambda(C_k)N_{\mu_k,\Sigma_k}(\vec{x}) \cdot (\vec{x}^i - \mu_C^i)(\vec{x}^j - \mu_C^j))d\vec{x}}{\sum_{k=1}^{m}(\int_{-\infty}^{+\infty} w_{C_k}\lambda(C_k)N_{\mu_k,\Sigma_k}(\vec{x}))d\vec{x})}$$

Let us note that we again need to employ a multiple integral to calculate the new covariance matrix because we do not have the actual data distribution at each site. Therefore, we assume that the local density given by each local clustering is a well enough description of this distribution. To consider the number of data objects that are stored at each site, we additionally weight the influence of each distribution with the cover we transmitted from this site.

The weight of C can be determined as

$$w_C = \frac{\sum_{C_i \in C} w_{C_i} \cdot \lambda(C_i)}{\sum_{C_i \in C} \lambda(C_i)}.$$

7.3.4 Scaling to High Dimensional Datasets

If we apply DMBC as proposed in the previous subsection on higher dimensional datasets ($d > 2$), we face the problem that, in order to compute both the mutual support as well as the covariance matrix of a merged cluster, we have to evaluate multiple integrals.

Thus, in order to be scalable for higher dimensional datasets, we propose a variant of DMBC that uses variances instead of covariances for cluster representation. In particular, we assume the attributes to be independent of each other and represent a cluster C by its mean vector μ_C and its d-dimensional variance vector ν_C. The i-th value of ν_C, denoted by ν_C^i, indicates the variance of the Gaussian along attribute i.

As a consequence, the resulting Gaussians form ellipsoid-shaped clusters that are constrained to be axis-parallel. We will see later in the experimental evaluation, that this simplification does not cause a significant loss of quality. However, the benefits of this modification are the following. First, we are able to solve the integral of the mutual support analytically. Second, to compute the variance vector of a merged cluster, we need to solve only one integral rather than multiple integrals. Third, the transfer cost for each local cluster are reduced from $O(d^2)$ for the covariance matrix to $O(d)$ for the variance vector.

In fact, the mutual support of a pair of clusters $C = \{C_1, C_2\}$ can be computed as

$$MS(C_1, C_2) = \prod_{i=1}^{d} \int_{-\infty}^{+\infty} N_{\mu_1^i, \nu_1^i}(\vec{x}^i) \cdot N_{\mu_2^i, \nu_2^i}(\vec{x}^i) d\vec{x}^i$$

where

$$N_{\mu^i, \nu^i}(\vec{x}^i) = \frac{1}{\sqrt{2\pi\nu^i}} \cdot e^{\frac{-(\vec{x}^i - \mu^i)^2}{2\nu^i}}.$$

The following lemma enables us to solve this integral over d-dimensions analytically.

Lemma 7.1 *Let $C_1 = (\mu_{C_1}, \nu_{C_1})$ and $C_2 = (\mu_{C_2}, \nu_{C_2})$ be local clusters. Then*

$$MS(C_1, C_2) = \prod_{i=1}^{d} \frac{1}{\sqrt{2\pi \cdot (\nu_1^i + \nu_2^i)}} \cdot \exp - \frac{(\mu_1^i - \mu_2^i)^2}{2 \cdot (\nu_1^i + \nu_2^i)}$$

Proof. *Let $\frac{1}{\sqrt{2\pi \cdot \nu_1^i}} \cdot \frac{1}{\sqrt{2\pi \cdot \nu_2^i}} \exp - \frac{(\mu_1^i - \vec{x}^i)^2}{2 \cdot \nu_1^i} - \frac{(\mu_2^i - \vec{x}^i)^2}{2 \cdot \nu_2^i} = \vartheta_i \cdot \frac{1}{\sqrt{2\pi \cdot \nu^i}} \cdot \exp - \frac{(\mu^i - \vec{x}^i)^2}{2 \cdot \nu^i}$. If we apply the logarithm to the equation, and replace μ^i and ν^i by:*

$$\mu^i = \frac{\mu_1^i \cdot \nu_2^i + \mu_2^i \cdot \nu_1^i}{\nu_2^i + \nu_1^i} \quad \text{and} \quad \nu^i = \frac{\nu_2^i \cdot \nu_1^i}{\nu_2^i + \nu_1^i},$$

it follows that

$$\vartheta_i = \frac{1}{\sqrt{2\pi} \cdot (\nu_1^i + \nu_2^i)} \cdot \exp - \frac{(\mu_1^i - \mu_2^i)^2}{2 \cdot (\nu_1^i + \nu_2^i)}$$

Thus, we obtain (cf. proof of Lemma 3.1)

$$MS(C_1, C_2) = \prod_{i=1}^{d} \int_{-\infty}^{+\infty} N_{\mu_1^i, \nu_1^i}(\vec{x}^i) \cdot N_{\mu_2^i, \nu_2^i}(\vec{x}^i) d\vec{x}^i =$$

$$\prod_{i=1}^{d} \int_{-\infty}^{+\infty} \sqrt{\nu_1^i \nu_2^i} \frac{1}{\sqrt{2\pi} \cdot \nu_1^i} \cdot \frac{1}{\sqrt{2\pi} \cdot \nu_2^i} \exp - \frac{(\mu_1^i - \vec{x}^i)^2}{2 \cdot \nu_1^i} - \frac{(\mu_2^i - \vec{x}^i)^2}{2 \cdot \nu_2^i} d\vec{x}^i =$$

$$\prod_{i=1}^{d} \int_{-\infty}^{+\infty} \sqrt{\nu_1^i \nu_2^i} \cdot \vartheta_i \cdot \sqrt{\frac{\nu_2^i + \nu_1^i}{\nu_2^i \nu_1^i}} N_{\mu^i, \nu^i}(\vec{x}^i) d\vec{x}^i =$$

$$\prod_{i=1}^{d} \frac{1}{\sqrt{2\pi \cdot (\nu_1^i + \nu_2^i)}} \cdot \exp - \frac{(\mu_1^i - \mu_2^i)^2}{2 \cdot (\nu_1^i + \nu_2^i)} \cdot \int_{-\infty}^{+\infty} N_{\mu^i, \nu^i}(\vec{x}^i) d\vec{x} =$$

$$\prod_{i=1}^{d} \frac{1}{\sqrt{2\pi \cdot (\nu_1^i + \nu_2^i)}} \cdot \exp - \frac{(\mu_1^i - \mu_2^i)^2}{2 \cdot (\nu_1^i + \nu_2^i)} \cdot 1.$$

□

The j-th component of the variance vector of the global cluster C which evolved from the merge of m clusters C_i is given as:

$$\nu_C^j = \sqrt{\frac{\int_{-\infty}^{+\infty} (\sum_{i=1}^{m} w_{C_i} \lambda(C_i) N_{\mu_i^j, \sigma_i^j}(\vec{x^j})) \cdot (\vec{x}^j - \mu_C^j)^2 d\vec{x^j}}{\sum_{i=1}^{m} (\int_{-\infty}^{+\infty} w_{C_i} \lambda(C_i) N_{\mu_i^j, \nu_i^j}(\vec{x^j}) d\vec{x^j})}}$$

Note that for component ν_C^j we compute only a 1-dimensional integral over dimension j.

7.4 Experimental Evaluation

We implemented our versions of DMBC in Java and run several tests on a workstation featuring two 1.8 GHz Opteron processors and 8 GByte main memory. The test bed consists of one artificial 2-dimensional dataset (denoted as DS1) and two real-word datasets (denoted as DS2 and DS3). The latter two are derived from 68,040 images of the corel image feature collection of the UCI KDD archive[NHBM98]. DS2 contains 9-dimensional color moments of images in HSV color space (mean, standard deviation and skewness). DS3 comprises a description of the corel images based on co-occurrence textures with 16 dimensions.

The experimental results of DMBC on the synthetic DS1 (Figure 7.3) demonstrates that our algorithm is capable to handle cases 1, 2(a)-(c) described in Section 7.3.1: DMBC finds the global cluster "A" the objects of which are existent on only one single client, i.e. client 2 (Case 1). DMBC finds the global clusters "B", "D", and "E" the objects of which are rather well described by local clusters on all sites (Case 2(a)). DMBC finds the global cluster "C" the objects of which are distrubuted over all sites such that none of the sites exhibit a local cluster (Case 2(c)). Several objects of clusters "D" and "E" are prototypes for Case 2(b) because they are members of local clusters that are built from objects predominantly belonging to another global cluster.

In order to demonstrate the robustness of our clustering algorithm w.r.t. the number of clusters on each client site, we performed distributed clustering with a predetermined number of clusters. We measured the agreement between the distributed clustering and the results of the centralized EM al-

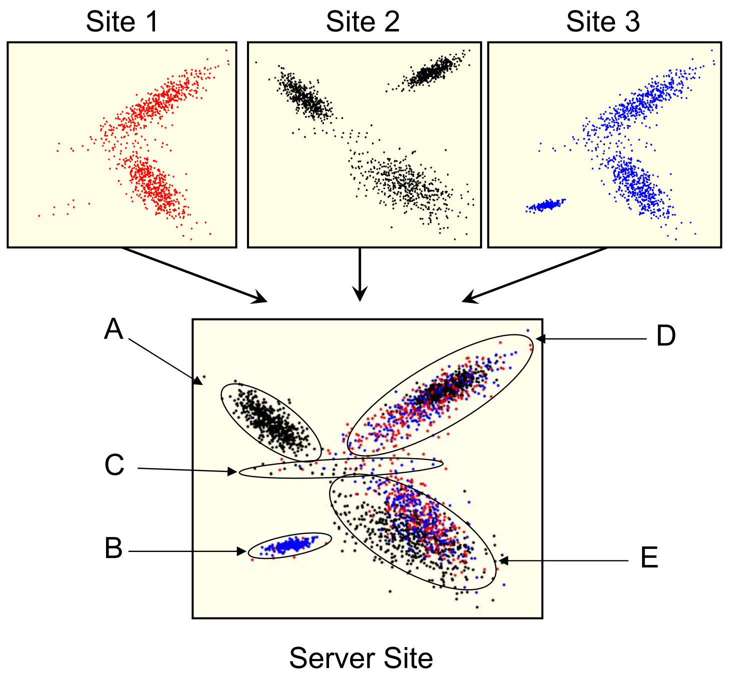


Figure 7.3: Results of DMBC on DS1.

gorithm using the Rand Index [HBV01], also known as Rand Statistics.

On the 2-dimensional synthetic dataset (DS1) shown in Figure 7.3 DMBC achieved a Rand index of approximately 99.9%, indicating a high agreement of our method with the centralized clustering. For data DS2 and DS3 we used the variant of DMBC based on variances (cf. Section 7.3.4). As shown in Figure 7.4, DMBC achieves high Rand Index values, i.e. our distributed approach produces a high level of agreement with the results of centralized clustering algorithms on all numbers of clusters. This also indicates that the variant proposed in Section 7.3.4 using variances instead of covariances does produce accurate results, too. We evaluated the scalability of the proposed algorithm w.r.t. the number of clusters on each client site. The results are depicted in Figure 7.4(a). As it can be observed, in all settings, the Rand Index is near the optimal value. Thus, the agreement between the centralized clustering and the global distributed clustering is very high.

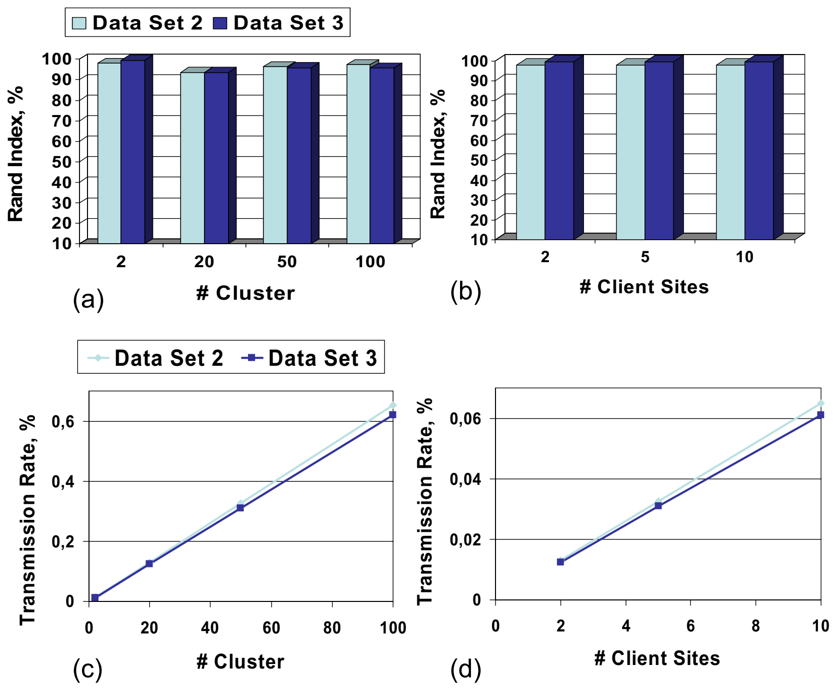


Figure 7.4: Results of DMBC on DS2 and DS3.

In addition, we investigated the scalability of the proposed algorithm w.r.t. the number of client sites. Figure 7.4(b) presents the agreement using the Rand Index between results calculated by our approach and the centralized clustering algorithm. The number of client sites involved in the distributed clustering was varying from 2 to 10. The high value of the Rand Index in all experiment evaluations shows that our algorithm is scalable w.r.t. the number of client sites and delivers results that do not differ from that of the global acting algorithm.

We also investigated the transfer cost w.r.t. the number of clusters on each client site and w.r.t. the number of client sites. The results are depicted in Figure 7.4(c) and 7.4(d). As transfer cost we measured the ratio of the number of bytes that are transfered using DMBC and of the number of bytes that are transfered using the centralized approach. As it can be seen, the transfer cost is in general very low. Even for a very large number of clusters, DMBC needs less than 1% of the bytes transfered by the centrailized approach. In addition, we can observe, that the transfer cost increases only linearly w.r.t. the number of local clusters and w.r.t. the number of client sites. Compared to other existing distributed clustering approaches, e.g. the density-based distributed approach in [JKP04], where the local transfer cost is at least 15% of the local data in order to achieve a high agreement, our DMBC reduces the transfer cost dramatically.

Last, we investigated the robustness of DMBC w.r.t. the probability threshold t which affects the cover of the local models. As the results (not shown due to space limitations) suggest, DMBC is rather robust w.r.t. a broad range of values for t. In fact, we observed a Rand Index over 98% when varying the values of t from 0.05 up to 0.45.

To sum up, our experiments demonstrated the robustness, the efficiency, and the applicability of both of our proposed variants for distributed model-based clustering.

7.5 Conclusions

In this chapter, we proposed a novel privacy-preserving clustering algorithm called distributed model-based clustering (DMBC). Our method can achieve an arbitrary level of privacy preservation by applying the EM algorithm at the local sites generating a model containing a set of Gaussian distributions. Each Gaussian is represented by its mean and its covariance matrix or — for higher dimensions the variance vector. We also proposed a merge step of the local Gaussians that can handle covariances as well as variances. Compared to recent approaches for pure distributed clustering, DMBC enables respecting an arbitrary level of privacy and dramatically reduces the transfer costs. Our experimental evaluation demonstrates the robustness, the efficiency, and the applicability of both of our proposed variants for distributed clustering.

Chapter 8

An EM-Approach for Clustering Multi-Instance Objects

Clustering multi-instance data is a very important but challenging task in mining advanced database systems as demonstrated in Chapter 1. This chapter introduces a novel model-based clustering algorithm for clustering multi-instance data. First, a short introduction into the clustering of multi-instance objects is given in Section 8.1. Section 8.2 surveys previous work in data mining multi-instance objects. Section 8.3 describes our statistical model for multi-instance data. In Section 8.4, this model is employed for EM clustering. To demonstrate the usefulness of the developed approach, Section 8.5 presents the results on several real-world datasets. Section 8.6 concludes the chapter with a summary.

8.1 Introduction

To cluster multi-instance (MI) objects, the common approach so far is to select some distance measures for point sets like [EM97, RB01] and then apply a distance-based clustering algorithm e.g. k-medoid methods like CLARANS [HK06] or a density-based algorithm like DBSCAN [EKSX96]. However, this approach does not yield expressive cluster models. Depending on the used algorithm, we might have some representative for some cluster, but we do not have a good model for describing the mechanism behind this clustering.

To overcome this problem, we will refer to the model of MI objects that was introduced in [WFP03] stating that a MI object of a particular class (or in our problem each cluster) needs to provide instances belonging to a certain concept or several concepts. We will adapt this view of MI objects to clustering. Therefore, we propose a statistical model that is based on 2 steps. In the first step, we use a standard EM Clustering algorithm on the union set of all MI objects. Thus, we determine a mixture model describing the instances of all MI objects. Assuming that each of the found clusters within each mixture model corresponds to some valid concept, we now can derive distributions for the clustering of MI objects. For this second step, we assume that a MI object containing k instances can be modeled as k draws from the mixture model over the instances. Thus, each cluster of MI objects is described by a distribution over the instance clusters derived in the first step and some prior probability.

8.2 Related Work

Data mining in multi-instance data objects has so far been predominantly examined in the classification section [DLLP97a, Zho04, WFP03, GFKS02a] as discussed in Section 2.2.3 of Chapter 2. To the best of our knowledge none of these approaches handles unsupervised learning or clustering.

For clustering multi-instance objects, it is possible to use distance functions for sets of objects like [EM97, RB01] (cf. Section 2.1.4 in Chapter 2 for details). Having such a distance measure, it is possible to cluster multi-instance objects with k-medoid methods like PAM and CLARANS [NH94] or employ density-based clustering approaches like DBSCAN [EKSX96]. Though this method yields the possibility to partition multi-instance objects into clusters, the clustering model consists of representative objects in the best case. Another problem of this approach is that the selection of a meaningful distance measure has an important impact of the resulting clustering. For example, netflow-distance [RB01] demands that all instances within two compared objects are somehow similar, whereas for the minimal Hausdorff [WZ00] distance the indication of similarity is only dependent on the closest pair.

In this chapter, we introduce an algorithm for clustering multi-instance objects that optimizes probability distributions to describe the dataset. Part of this work is based on expectation maximization (EM) clustering for ordinary feature vectors using Gaussians. Details about this algorithm can be found in Chapter 2.

8.3 A Statistical Model for Multi-Instance Objects

In this section, we will introduce our model for multi-instance clustering. Therefore, we will first of all define the terms instance and multi-instance (MI) object.

Definition 8.1 (instance and MI object)

Let F be a feature space. Then, $i \in F$ is called an instance in F. A multi-

instance (MI) object o in F is given by an arbitrary sized set of instances $o = i_1, .., i_k$ with $i_j \in F$. To denote the unique MI object an instance i belongs to, we will write $MiObj(i)$.

To cluster multi-instance objects using an EM approach, we first of all need a statistical process that models sets of multi-instance objects. Since multi-instance objects consist of single instances in some feature space, we begin with modeling the data distribution in the feature space of instances. Therefore, we first of all define the instance set of a set of multi-instance objects:

Definition 8.2 (Instance Set)
Given a database DB of multi-instance Objects $o = i_1, \ldots, i_k$, the corresponding instance set $I_{DB} = \bigcup_{DB} o$ is the union of all multi-instance objects.

To model the data distribution in the instance space, we assume a mixture model of k independent statistical processes. For example, an instance set consisting of feature vectors could be described by a mixture of Gaussians.

Definition 8.3 (Instance Model)
Let DB be a dataset consisting of multi-instance objects o and let I_{DB} be its instance set. Then, an instance model IM for DB is given by a mixture model of k statistical processes that can be described by a prior probability $Pr[k_j]$ for each component k_j and the necessary parameters for the process corresponding to k_j, e.g. a mean vector μ_j and co-variance matrix M_j for Gaussian processes.

After describing the instance set, we can now turn to the description of multi-instance objects. Our solution is based on the idea of modeling a cluster of multi-instance objects as a multinomial distribution over the components of the mixture model of instances. For each instance and each concept, the probability that the instance belongs to this concept is considered as result of one draw. If the number n of instances within an object o is considered to be important as well, we can integrate this into our model as well by considering some distribution over the number of draws, e.g. a binomial distribution. To conclude, a mixture model of multi-instance clusters can be described by a set of multinomial distributions over the components of a mixture model of instances. A multi-instance object is thus derived in the following way:

1. Select a multi-instance cluster c_i w.r.t. some prior distribution over the set of all clusters C.
2. Derive the number of instances n within the multi-instance object w.r.t some distribution depending on the chosen cluster c_i.
3. Repeat n-times:
   (a) Select some model component k_j within the mixture model of instances w.r.t. the multi-instance cluster specific distribution.
   (b) Generate an instance, w.r.t. to the distribution corresponding to component k_j.

Formally, the underlying model for multi-instance datasets can be defined as follows:

Definition 8.4 (Multi-Instance Model)
A multi-instance model M over the instance model IM is defined by a set C of l processes over I_{DB}. Each of these processes c_i is described by a prior probability $Pr[c_i]$, a distribution over the number of instances in the bag $Pr[Card(o)\ |c_i]$ and an conditional probability describing the likelihood that a multi-instance object o belonging to process c_i contains an instance belonging to the component $k_l \in IM$. The probability of an object o in the model M is calculated as following:

$$Pr[o] = \sum_{c_i \in C} Pr[c_i] \cdot \ Pr[Card(o)|c_i] \cdot \prod_{i \in o} \prod_{k \in MI} Pr[k|c_i]^{Pr[k|i]}$$

The conditional probability of process c_i under the condition of a given multi-instance object o can be calculated by:

$$Pr[c_i|o] = \frac{1}{Pr[o]} \cdot Pr[c_i] \cdot \ Pr[Card(o)|c_i] \cdot \prod_{i \in o} \prod_{k \in MI} Pr[k|c_i]^{Pr[k|i]}$$

Let us note that the occurrence of an instance within the data object is only dependent on the cluster of instances it is derived from. Thus, we do not assume any dependencies between the instances of the same objects. Another important characteristic of the model is that we assume the same set of instance clusters for all multi-instance clusters. This assumption leads to the following 3 step approach for multi-instance EM clustering.

8.4 EM-Clustering for Multi-Instance Objects

After introducing a general statistical process for multi-instance objects, we will now introduce an EM algorithm that fits the distribution parameters to a given set of multi-instance objects. Our method works in 3 steps:

1. Derive a Mixture Model for the Instance Set.
2. Calculate a start partitioning.
3. Use the new EM algorithm to optimize the start partitioning.

8.4.1 Generating a Mixture Model for the Instance Set

To find a mixture of the instance space, we can employ a standard EM approach as proposed in Chapter 2. For general feature vectors, we can describe the instance set as a mixture of Gaussians. If the feature space is sparse using a mixture of multinomial processes usually provides better results. If the number of clusters in the instance is already known, we can simply employ

EM clustering. However, if we do not know how many clusters are hidden within the instance set, we need to employ a method for determining a suitable number of processes like [Smy96].

8.4.2 Finding a Start Partitioning of Multi-Instance Objects

After deriving a description of the instance space, we now determine a good start partitioning for the final clustering step. A good start partitioning is very important for finding a good cluster model. Since EM algorithms usually do not achieve a global maximum likelihood, a suitable start partitioning has an important impact on both, the likelihood of the cluster and the runtime of the algorithm. The versions for EM in ordinary feature spaces often use k-means clustering for finding a suitable start partitioning. However, since we cluster sets of instances instead of single instances, we cannot use this approach directly.

To overcome this problem, we proceed as follows. For each multi-instance object we determine a so-called confidence summary vector in the following way.

Definition 8.5 (Confidence Summary Vector)
Let IM be an instance model over database DB containing k processes and let o be a multi-instance object. Then the confidence summary vector $\overrightarrow{csv}(o)$ of o is a k dimensional vector that is calculated as follows:

$$csv_j(o) = \sum_{i \in o} Pr[k_j] \cdot Pr[i|k_j]$$

After building the confidence summary vector for each object, we can now employ k-means to cluster the multi-instance objects. Though the resulting clustering might not be optimal, the objects within one cluster should yield similar distributions over the components of the underlying instance model.

8.4.3 EM for Clustering Multi-Instance Objects

In this final step, the start partitioning for the dataset is optimized using the EM algorithm. We therefore describe a suitable expectation and maximization step and then employ an iterative method. The likelihood of the complete model M can be calculated by adding up the log-likelihoods of the occurrence of each data object in each clusters. Thus, our model is (locally) optimal if we obtain a maximum for the the following log-likelihood term.

Definition 8.6 (Log-Likelihood for M)

$$E(M) = \sum_{o \in \mathcal{DB}} \log \sum_{c_i \in M} Pr[c_i|o]$$

To determine $Pr[c_i|o]$, we proceed as mentioned in definition 8.4. Thus, we can easily calculate $E(M)$ in the expectation step for a given set of distribution parameters and an instance model. To improve the distribution parameters, we employ the following updates to the distribution parameters in the maximization step:

$$W_{c_i} = Pr[c_i] = \frac{1}{Card(DB)} \sum_{o \in \mathcal{DB}} Pr[c_i|o]$$

where W_{c_i} denotes the prior probability of a cluster of multi-instance objects.

To estimate the number of instances contained in an MI object belonging to cluster c_i, we can employ a binomial distribution determined by the parameter l_{c_i}. The parameters are updated as follows:

$$l_{c_i} = \frac{\sum_{o \in \mathcal{DB}} Pr[c_i|o] \cdot Card(o)}{Card(DB)} \cdot \frac{1}{MAXLENGTH}$$

where $MAXLENGTH$ is the maximum number of instances for any MI object in the database.

Finally, to estimate the relative number of instances drawn from concept k_j for MI objects belonging to cluster c_i, we derive the parameter updates in

	Dataset 1 (DS1)	**Dataset 2 (DS2)**	**Dataset 3 (DS3)**
Name	Brenda	MUSK 1	MUSK 2
Number of MI-Objects	6082	92	102
Average Number of Instances per MI-Object	1.977	5.2	64.7
Number of MI-Object classes	6	2	2

Table 8.1: Details of the test environments.

the following way:

$$P_{k_j,c_i} = Pr[k_j|c_i] = \frac{\sum_{o\in\mathcal{DB}} (Pr[c_i|o] \cdot \sum_{u\in o} Pr[u|k_j])}{\sum_{o\in\mathcal{DB}} Pr[c_i|o]}$$

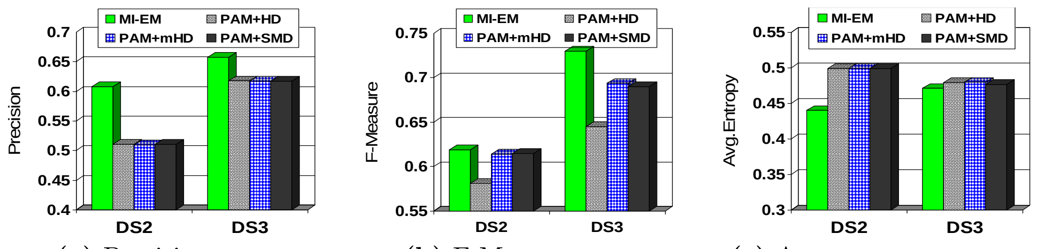
Using these update steps, the algorithm is terminated after the improvement of $E(M)$ is less than a given value σ. Since the last step of our algorithm is a modification of EM clustering based on multinomial processes, our algorithm always converges against a local maximum value for $E(M)$.

8.5 Experimental Evaluation

All algorithms are implemented in Java 1.5. The experiments described below are carried out on a work station that is equipped with two 1.8 GHz Opteron processors and 8 GB main memory.

Our experiments were performed on 3 different real-world datasets. The properties of each test bed are illustrated in Table 8.1. The Brenda dataset contains of enzymes taken from the protein data bank (PDB) [1]. Each enzyme comprises several chains given by amino acid sequences. In order to derive feature vectors from the amino acid sequences, we employed the approach

[1]http://www.rcsb.org/pdb/



(a) Precision. **(b)** F-Measure. **(c)** Average entropy.

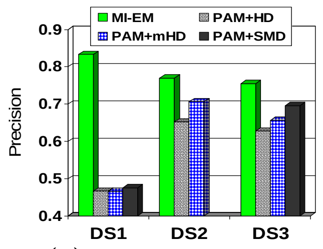
Figure 8.1: Effectiveness evaluation on DS2 and DS3 (no. of clusters is 2).

described in [WMSW01]. The basic idea is to use local (20 amino acids) and global (6 exchange groups) characterization of amino acid sequences. In order to construct a meaningful feature space, we formed all possible 1-grams for each kind of characteristic. This approach provided us with 26 dimensional histograms for each chain. To obtain the class labels for each enzyme we used a mapping from PDB to the enzyme class numbers from the comprehensive enzyme information system BRENDA [2].

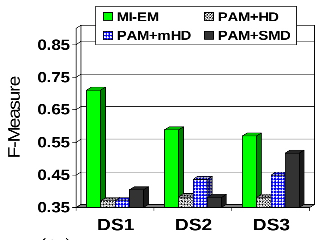
MUSK 1 and MUSK 2 datasets come from UCI repository [NHBM98] and describe a set of molecules. The MI-objects in MUSK 1 and MUSK 2 datasets are judged by human experts to be in musks or non-musks class. The feature vectors of MUSK datasets have 166 numerical attributes that describe these molecules depending on the exact shape or conformation of the molecule.

In order to demonstrate that the proposed clustering approach for multi-instance objects outperforms standard clustering algorithms working on a suitable distance functions, we compared precision, F-Measure and average entropy of the MI-EM with that of k-medoid clustering algorithm (PAM). To enable cluster analysis of multi-instance objects by PAM, we used the Hausdorff distance (HD)[EM97], the minimum Hausdorff distance (mHD)[WZ00] and the Sum of Minimum Distances (SMD)[EM97]. Due to the fact that

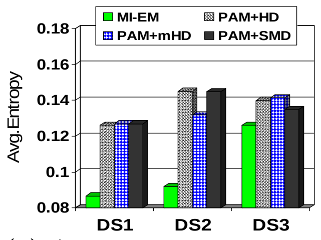
[2]http://www.brenda.uni-koeln.de/



(a) Precision.



(b) F-Measure.

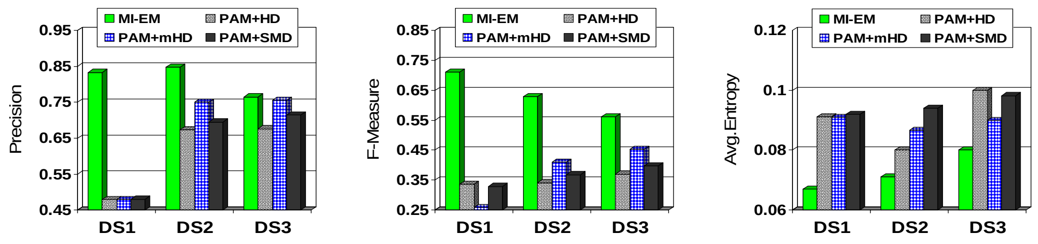


(c) Average entropy.

Figure 8.2: Effectiveness evaluation on DS1, DS2 and DS3 where no. of clusters is 6.

the dataset DS1 has 6 classes and the datasets DS2 and DS3 have 2 classes, we investigated the effectiveness of the cluster analysis where the number of clusters is equal to or slightly than the number of the desired classes. Thus, we set in our experiments the number of clusters equal to 6 and 8 for DS1, and equal to 2, 6 and 8 for the datasets DS2 and DS3. The results of our comparison are illustrated in Figures 8.1,8.2 and 8.3.

In all our experiments, PAM working on distance functions suitable for multi-instance objects achieved a significantly lower precision than MI-EM. For example, the MI-EM algorithm reached a precision of 0.833 on DS1 and the number of clusters equal to 8 (cf. Figure 8.3(a)). In contrast to the result of MI-EM, the precision calculated for clusterings found by all competitors lies between 0.478 and 0.48. Furthermore, MI-EM obtained in all experiments higher or comparable values of F-Measures. This fact indicates that the cluster structure found by applying of the proposed EM-based approach is more exact w.r.t. precision and recall than that found by PAM with 3 different MI distance functions. For example, the F-Measure calculated for MI-EM clustering of DS2 with 8 clusters is 0.63 whereas PAM clustering with different MI distance functions shows values between 0.341 and 0.41 (cf. Figure 8.3(b)). Finally, the values of average entropy observed by the MI-EM results are considerably lower than those of PAM on HD,



(a) Precision. **(b)** F-Measure. **(c)** Average entropy.

Figure 8.3: Effectiveness evaluation on DS1, DS2 and DS3 where no. of clusters is 8.

mHD and SMD. The lower values of average entropy imply a lower level of impurity in the cluster structures detected by applying MI-EM.

To summarize, the values of the different quality measures observed on real-world datasets when varying the number of clusters show that the proposed EM-based approach for cluster analysis of MI-objects outperforms the considered competitors w.r.t. effectiveness.

8.6 Conclusions

In this chapter, we described an approach for statistical clustering of multi-instance objects. Our approach models instances as members of concepts in some underlying feature space. Each concept is modeled by a statistical process in this feature space, e.g. a Gaussian. A multi-instance object can now be considered as the result of selecting several times a concept and generating an instance with the corresponding process. Clusters of multi-instance objects can now be described as multinomial distributions over the concepts. In other words, different clusters are described by having different probabilities for the underlying concepts. An additional aspect is the length of the MI object. To derive multi-instance clusters corresponding to this model, we introduce a three step approach. In the first step we derive a mixture model describing concepts in the instance space. The second step finds a good initialization for the target distribution by subsuming each multi-instance object by a so-called confidence summary vector (csv) and afterwards clustering these csvs using the k-means method. In the final, step we employ a final EM clustering step optimizing the distribution for each cluster of multi-instance objects. To evaluate our method, we compared our clustering approach to clustering multi-instance objects with the k-medoid clustering algorithm PAM for 3 different similarity measures. The results demonstrate that the found clustering model offers better cluster qualities w.r.t. to the provided reference clusterings.

Chapter 9

Conceptually Specified Multi-Instance Clusters

Recently, more and more applications represent data objects as sets of feature vectors or multi-instance objects as discussed in Chapter 1. In this chapter, we propose a new method for clustering multi-instance objects, called COSMIC (**CO**nceptually **S**pecified **M**ulti-**I**nstance **C**lusters). We start with a motivation of multi-instance clustering in Section 9.1. Afterwards, Section 9.2 surveys previous work in data mining with multi-instance objects. Furthermore, we will provide a short introduction to OPTICS which is the foundation of our method to derive concept hierarchies. Section 9.3 provides the necessary formal framework for our approach to clustering multi-instance objects. Then, Section 9.4 describes the complete COSMIC algorithm for generating a concept lattice. Section 9.5 displays the results of our experimental evaluation, and Section 9.6 concludes the chapter with a summary.

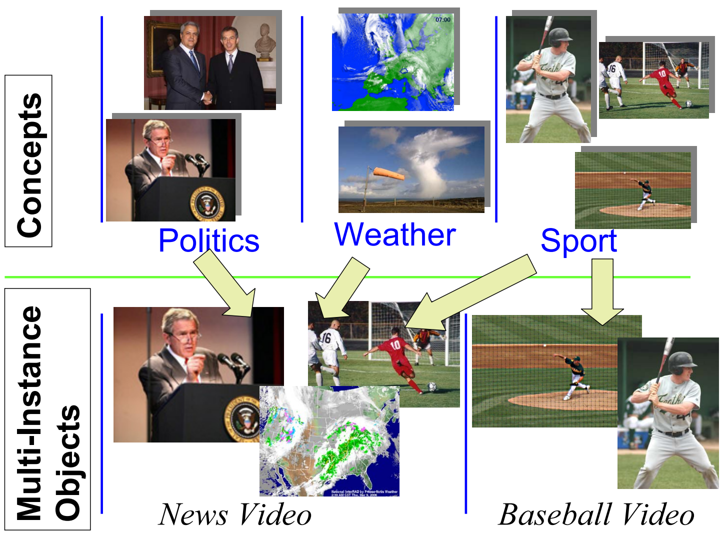


Figure 9.1: Example of a Multi-Instance object: a video clip as a set of scene concepts.

9.1 Introduction

In advanced data mining applications, the complexity of the data objects is increasing as rapidly as their plain number. Therefore, more and more data mining applications employ sets of feature vectors or multi-instance objects to represent a single data object. For example, a molecule can be described by the set of all conformations or shapes it might adopt [DLLP97b]. Other examples are websites where each site can be considered as a set of webpages as described in [EKS02], and CAD-parts which are decomposed into several spatial primitives or covers as in [BKK$^+$03]. In this chapter, we will encounter another application by decomposing a protein into a set of subunits, i.e. each strand of amino acids that builds the protein is considered separately. One final example for using sets of feature vectors is the clustering of video clips. A video can be considered as a sequence of scenes. If the order of the scenes is irrelevant, the video clip is represented by a set of scene descriptors. An example for such a setting is the clustering of news clips taken from different tv-stations. Though the order of the news might be varying, all stations will broadcast similar scenes of the current top stories (cf. Figure 9.1).

In general, a multi-instance object is represented by a set of object descriptions of one and the same type, e.g. color histograms or text vectors. We will call the elements of a multi-instance object *instances*. Each instance can describe a different view of the complete object like in [DLLP97b] or a different part of the complete object like in [BKK$^+$03, EKS02].

The main direction for clustering multi-instance objects so far was the development of distance measures for set-valued objects like [EM97, RB01]. Given a distance measure, standard clustering algorithms like DBSCAN [EKSX96], OPTICS [ABKS99], or k-medoid clustering [HK06] are applicable to cluster multi-instance objects. Though distance measures for multi-instance objects yield a solution for the given problem, they also have serious drawbacks. The selection of the right distance function has a great impact on the success of clustering. Additionally, the resulting clusters are often hard to interpret.

In this chapter, we propose COSMIC, a method for deriving (**CO**nceptually **S**pecified **M**ulti-**I**nstance **C**lusters). The idea of COSMIC is to derive a so-called concept lattice as known from formal concept analyzes [GW99] to describe the rich relationships between sets of feature vectors. Based on the concept lattice we can derive flat as well as hierarchical clusterings.

In formal concept analysis, each object o can be described by a set of nominal attributes $Desc(o)$. A concept C is now defined by a set of objects and a set of attributes $Desc(C)$ if for each object $o \in C$, $Desc(C) \subseteq Desc(o)$ holds. In other words, a concept is the maximal set of objects that can be described by $Desc(C)$. Let us note that it is possible that the elements of a concept C contain additional attributes and thus, concepts can overlap, i.e. one object can be contained in multiple concepts. Additionally, concepts can be specialized and generalized to sub- or super concepts by adding or dropping attributes which decreases or increases the set of objects that are covered by the concept. Therefore, the set of all concepts for a given set of objects and attributes is organized in the so-called concept lattice.

Having a concept lattice instead of a plain clustering yields much more information about the contained patterns. Concepts can be interpreted easily by the given concept description. Overlapping concepts naturally describe the characteristics of sets because some multi-instance object o_1 might be similar to another multi-instance object o_2 w.r.t. a certain subset of its instances and also be similar to another multi-instance object o_3 w.r.t. to another subset. For example, when considering web sites as sets of web pages, a site containing “disclaimer page”, “job vacancies”, “business reports”, and “pharmacy products” could be assigned to a cluster of sites which is described by “disclaimer pages”, “job vacancies”, and “business reports” in one sense. Additionally, the same site should be assigned to a cluster of web sites containing “job vacancy” and “pharmacy products” pages. A concept lattice expresses these multiple relationships in a natural way.

To derive a concept lattice from a set of multi-instance objects, the key issue is to describe each object by a set of attributes drawn from the corresponding instances. Thus, we have to find groups of instances having a similar meaning. One obvious solution to this problem is to cluster the instances. Each cluster of instances does now provide a so-called attribute and an multi-instance object can be described by the set of clusters its instances belong to. We will refer to the attributes describing clusters as *concept attributes* (CA) to distinguish them from the numerical attributes spanning the instance feature space. However, in order to subsume the instances of a multi-instance object by a set of meaningful attributes, the employed clustering algorithm has to cope with several demanding challenges.

The clustering algorithm should only group objects into a cluster that are really similar. Otherwise, multi-instance objects might be described by so-called phantom attributes. A phantom attribute is caused by an instance that is assigned to a cluster that does not really describe its content. As a result the corresponding object description becomes misleading. Let us note that this is especially a problem of partitioning clustering algorithms because in this approach each instance has to be assigned to some cluster regardless whether there are any similar instances in the dataset at all. Therefore, the employed clustering algorithms should rather skip these instances or assign a noise label instead of generating phantom attributes. Another problem of many partitioning methods is that it is necessary to specify the number of clusters to be found in the dataset. A too small number might lead to phantom attributes because dissimilar instances have to be packed into the same clusters. A too large number will lead to so-called duplicate attributes that actually describe the same type of instance. Thus, very similar multi-instance objects might not be recognized. An effect further escalating this problem is that the question of disjunctive attributes or joined attributes also depends on the level of abstraction. In other words, in some applications it would make sense to separate the concepts "scientific job pages" and "administrative job pages". In other applications it would be more useful to consider the more general attribute "job pages". Another important challenge is to prevent the clustering algorithms to derive useless clusters corresponding to attributes that only describe a too small number of objects. However, this can easily happen if each multi-instance object contains a large number of very similar instances which are naturally grouped into the same cluster. As a result, we might end up with attributes like "page that was found in www.dbs.ifi.lmu.de".

To cope with all these problems, COSMIC relies on the hierarchical density-based notion of clustering which offers suitable solutions to the named problems. Density-based clustering as proposed in [ABKS99, EKSX96] can automatically detect the number of clusters in a given dataset and assigns a "noise" label to instances that are not similar enough to a sufficiently large subset of other instances. Thus, we prevent the generation of phantom and duplicate attributes. However, since we additionally need to consider the relationships between the derived attributes, we generate a hierarchical clustering providing an attribute hierarchy. This way, similar attributes might be subsumed into a more general attribute on a higher abstraction level. To cope with the problem of meaningless attributes, COSMIC is based on a modified version of reachability distance compared to general hierarchical density-based clustering [ABKS99].

9.2 Related Work

Data mining in multi-instance data objects has so far been predominantly studied w.r.t. classification. In [DLLP97b], Dietterich et al. defined the problem of multi-instance learning for drug prediction and provided a specialized algorithm to solve this particular task (see Section 2.2.3 for details). In the following years, new algorithms for this rather specialized task were introduced (for a survey cf. [Zho04]). Since the methods for this approach are limited to the case that there is only one single *concept attribute* (CA) and relevant multi-instance objects carry at least one instance belonging to this CA, [WFP03] introduced a more general method for handling multi-instance objects. This model considers several CAs for each class and defines classes by multiple CAs that must occur in a certain cardinality. However, all of these approaches are supervised because they require sets of labeled MI objects.

Additionally, multi-instance objects were handled by complex distance measures [EM97, RB01] or kernel functions [GFKS02a]. Employing these similarity measures, it is possible to employ distance-based data mining approaches like k-NN classification, k-medoid Clustering [HK06] or OPTICS [ABKS99], or kernel methods [GFKS02a]. However, the selection of a suitable similarity measure for a particular application is often quite difficult and the proposed similarity measures for multi-instance objects often vary strongly when measuring the similarity between multi-instance objects. Therefore, it is often necessary to try out a large variety of parameters and distance measures. Another problem is the tangibility of the derived clusters. For complex similarity measures and large MI objects containing hundreds of instances, it is very difficult to understand why the multi-instance objects belonging to the same cluster are considered to be similar. Finally, employing complex distance measures often leads to efficiency problems. Since a considerable part of the similarity measures for multi-instance objects is non-metric, employing index structures is not always possible. Additionally, useful filter steps avoiding time consuming distance calculations as in [BKK$^+$03] were introduced for a minority of multi-instance distance measures only.

Clustering data objects based on a concept lattice was previously used for other data types such as text data [SC99]. However, to the best of our knowledge none of these methods deals with multi-instance objects and the question of how to derive CAs from a set of multi-instance objects.

Our method adapts the OPTICS algorithm [ABKS99] to derive CAs. The OPTICS algorithm derives a cluster hierarchy that is displayed within the so-called reachability plot (see 2.2.1. To derive a cluster hierarchy, several methods have been proposed [ABKS99, SQL$^+$03, BKKP04] which extract the hierarchy of all occurring clusters based on several input parameters. In contrast to these methods, COSMIC does not employ any parameters to decide whether or not a cluster should be extracted. Instead, the usefulness for describing a further concept determines the existence of an instance cluster.

9.3 Preliminaries

In this section, we will formalize multi-instance (MI) objects and concept lattices which are the result of COSMIC. As mentioned before, an MI object is a set of object descriptions called instances. For example, a web site is an MI object and its instances are the web pages within this site. Formally, we can define an MI object as follows:

Definition 9.1 (Instance and MI object)
Let F be a feature space. Then, $i \in F$ is called an instance *in F. A* multi-instance (MI) object *o is given by an arbitrary sized set of instances $o = \{i_1, \ldots, i_k\}$ where $i_j \in F$. To denote the unique MI object an instance i belongs to, we will write MiObj(i).*

To derive a concept lattice, we need to transform MI objects to objects that are described by a set of nominal attributes. Thus, we employ clustering to group several instances to so-called concept attributes:

Definition 9.2 (Concept Attribute (CA))
Let F be a feature space. A concept attribute (CA) *c describes a set of similar instances $\mathcal{I}_c \subset F$. For any $i \in \mathcal{I}_c$, we will denote ConAttr$(i) = c$.*

Each CA c can now be considered as a nominal attribute describing each MI object containing at least one element of $\mathcal{I}_c$. As mentioned above, we consider the CAs to be organized in a hierarchy. Thus, a CA might generalize several more specialized CAs. Consider for example the CA "product descriptions". Subconcept Attributes (SubCAs) might be "descriptions of hardware products" and "descriptions of software products". A CA s is called *direct* SubCA of c if there is no other SubCA t of c such that s is also a SubCA of t. We formalize this idea in the relation SubCA:

Definition 9.3 (SubConcept Attribute (SubCA))
Let s, c be two concept atttibutes in F where $\mathcal{I}_s \subseteq F$ and $\mathcal{I}_c \subseteq F$ are the sets of members of s and c, respectively. Then, s is called subconcept attribute *(SubCA) of c, denoted by $SubConAttr_c(s)$ if $\mathcal{I}_s \subset \mathcal{I}_c$. Additionally, s is called* direct subconcept attribute *of c iff*

$$\nexists r : SubConAttr_c(r) \wedge SubConAttr_r(s)$$

To define a hierarchy, we start with one root CA c_{all} containing all instances in F. Then, all CAs except for the root CA are a SubCA of at least one other CA. More formally:

Definition 9.4 (Concept Attribute Hierarchy)
Let $H = \{c_1, ..., c_n\}$ be a set of concept attributes in F. H is called a concept hierarchy if the following conditions hold:

$$\begin{array}{ll} (1) & \exists c_{all} \in H \ : \forall i \in F : i \in \mathcal{I}_{c_{all}} \\ (2) & \forall c_i \in H \backslash c_{all} \, \exists c_j \in H : SubConAttr_{c_j}(c_i) \end{array}$$

Having derived a mapping of instances to CAs, we now will formalize the resulting concept lattice as introduced in formal concept analysis. Therefore, we will first of all introduce a formal context (similar to [GW99]) to specify the complete set of objects, the describing attributes and which object is described by which attribute:

Definition 9.5 (Formal Hierarchical Context)
Let $\mathcal{DB}$ be a set of objects and let H be a CA hierarchy. A formal hierarchical context is now given by the triple (O, H, I) where I is a binary relation between O and H: $I \subseteq (O \times H)$, and the following condition holds:

$$\begin{array}{l} \forall c_i, c_j \in H, \forall o \in \mathcal{DB} : \\ \quad SubConAttr_{c_j}(c_i) \wedge (o, c_i) \in I \Rightarrow (o, c_j) \in I. \end{array}$$

Thus, the context defines which CAs are contained in which object. Furthermore, the condition states that if an object is described by a CA c_i, then it must also be described by all of the ancestors of c_i in the CA hierarchy.

To describe the output of COSMIC, we first of all need to specify a single concept:

Definition 9.6 (Concept)
Let $(\mathcal{DB}, H, I)$ be a formal hierarchical context. An object set $C \subseteq \mathcal{DB}$ together with a CA set $Desc(C) \subseteq H$ is called concept if the following conditions hold:

$$\begin{aligned} &(1) \quad C = \{o \in \mathcal{DB} | \forall a \in H : (o,a) \in I\} \\ &(2) \quad Desc(C) = \{a \in H | \forall o \in \mathcal{DB} : (o,a) \in I\} \end{aligned}$$

We will call $Desc(C)$ the concept description of C.

Since a concept that is not general enough is not useful for examining patterns in a dataset, we will call the cardinality of a concept C the support of C, denoted by *support*(C). For building a concept lattice, it is therefore often sufficient to only consider concepts that have a support above a certain minimum threshold *MinSup*.

In other words, a concept is the maximal subset of the objects which contains elements of the concept description $Desc(C)$. For example, a concept of websites could be described by the concept attributes "employment", "financial reports", and "software development". Each website belonging to the concept must contain at least one web page belonging to each of these CAs. At the same time there is no website in the given context being described by these concept attributes that is not part of the concept. To describe the relationship between two concepts, we will now specify the subconcept:

Definition 9.7 (Subconcept)
Let C_1, C_2 be two concepts in $\mathcal{DB}$ w.r.t. to the context $(\mathcal{DB}, H, I)$. Then C_1

is called subconcept *of C_2, denoted by $SubConcept_{C_2}(C_1)$ iff*

$$C_1 \subset C_2 \quad \Leftrightarrow \quad Desc(C_1) \supset Desc(C_2)$$

In other words, a subconcept C_2 of concept C_1 contains only a subset of the objects in C_1. Additionally, the concept description must contain at least one additional attribute that is not contained in the description of the father concept. For example, a concept of web sites which is described by the CAs "faculty pages" and "lectures" is a generalization of the subconcept being described by "faculty pages", "lectures" and the additional CA "computer science lectures".

Finally, we can specify the concept lattice for a given context:

Definition 9.8 (Concept Lattice)
Let $(\mathcal{DB}, H, C)$ be a formal context. The set of all concepts that can be found in the context $(\mathcal{DB}, H, C)$ together with the subconcept relation between these concepts is called concept lattice.

The resulting concept lattice now describes an overlapping hierarchical grouping of an MI dataset that can be explored directly. Additionally, we can use the concept lattice to derive a flat disjunctive clustering by assigning each MI object to the most specialized concept it belongs to.

The goal of COSMIC is to derive a concept lattice over a dataset of MI objects containing all concepts having at least a support of *MinSup*. The CAs this lattice is based on are organized in a hierarchy and it is guaranteed that each CA in this hierarchy is employed to describe at least one cluster. While processing, COSMIC avoids considering useless candidates for CAs whenever possible. Hence, COSMIC is rather efficient.

9.4 COSMIC

In this section, we will introduce COSMIC, our new approach to derive concept lattices from MI objects. Therefore, we will start by giving a general description of our algorithm. Subsequently, we will describe the two main steps of our method in more detail.

The input of COSMIC is a set of MI objects over an arbitrary feature space F. Additionally, we need a distance measure $dist : F \times F \rightarrow \mathbb{R}^+$ for comparing the instances. The result of COSMIC is a concept lattice which is defined on the basis of a CA hierarchy. COSMIC proceeds in two steps:

1. The first step determines a cluster order representing a cluster hierarchy containing all potential CAs.
2. The second step derives a concept lattice and a hierarchy of CAs that are used to describe these clusters.

The first step is based on the density-based notion of clustering and, thus, needs the parameters which are specific for this approach, i.e. *MinPts* and ε. In the second step, we only need to specify the minimum support of the concepts called *MinSup*.

9.4.1 Deriving a Concept Hierarchy

The first step of COSMIC aims at the construction of an expressive cluster hierarchy of instances which provides candidates for CAs. The clusters are only candidates since we will only consider a cluster to be a CA if it is contained in the description of at least one concept. Our approach is based on the density-based hierarchical clustering algorithm OPTICS [ABKS99]. Though OPTICS derives rather a reachibility plot than a real cluster hierarchy, it yields several advantages. The algorithm is very robust w.r.t. its two parameters ε and *MinPts*. Specifying the ε-parameter in OPTICS is more or less only necessary when employing index structures for efficiently processing ε-range queries. If no index structure is applicable, the ε-parameter can always be set to infinity to guarantee that all clusters for the given *MinPts* are found. Thus, specifying ε is rather a question of performance tuning than a question of clustering quality. The result of OPTICS is mostly dependent on the *MinPts* parameter which controls how many other instances have to be found in a local neighborhood to indicate a dense area in the dataset. The value of *MinPts* controls the smoothness of the resulting reachability. If *MinPts* is too small, even small variations in local density will cause the existence of new clusters. Though OPTICS provides a meaningful description of the cluster structure in the instance space, the clusters derived by OPTICS are often not suitable for describing a CA hierarchy.

The reason for this problem is that OPTICS does not distinguish between instances belonging to the same MI object and instances that were taken from different MI objects. In the following, we will motivate the problem with an example, we encountered during our experiments. When clustering the pages of a website dataset using OPTICS, we obtained several clusters. However, a closer investigation of the generated clusters indicated that the web pages within one cluster often belonged to a single website. Similar observations were made on datasets describing molecules as sets of their conformations. Obviously the similarity between the instances of the same MI object was considerable higher than the similarity to the instances belonging to any other MI object. This is a problem because a CA describing a single MI object is useless to describe an MI cluster, i.e. a group of MI objects. For example, the candidate CA "page of www.lmu.de" is unlikely to describe a web page in any other website than "www.lmu.de". Besides the problem that these clusters are useless for describing concepts, allowing this type of clusters often prevents the detection of useful clusters which are capable to describe similar groups of instances taken from various MI objects.

A solution to this problem needs to make sure that each cluster contains *MinSup* instances belonging to at least *MinSup* different MI objects. Thus, the cluster order is guaranteed to exclusively contain clusters that are at least potentially useful CAs describing at least one concept with the minimum support *MinSup*. To integrate this requirement into density-based clustering, we redefine the core-distance of OPTICS into the *concept core-distance*:

Definition 9.9 (Concept core-distance)
Let MinPts $\in \mathbb{N}$, $\varepsilon \in \mathbb{R}^+$ *and let* $\mathcal{DB}$ *be a set of MI objects and* $\mathcal{I} = \bigcup_{o \in \mathcal{DB}} o$. *The MinPts-nearest MI neighbors of an instance* i *are the smallest set* $N^{MI}_{MinPts}(i) \subseteq \mathcal{I}$ *that contains (at least) MinPts instances for which the following conditions hold:*

$$\begin{aligned}
&(1) \quad \forall p \in N^{MI}_{MinPts}(i), \forall q \in \mathcal{DB} \setminus N^{MI}_{MinPts}(i) : \\
&\qquad dist(p,i) < dist(q,i) \\
&(2) \quad |\{MiObj(x) | x \in N^{MI}_{MinPts}(o)\}| \geq MinPts
\end{aligned}$$

Then, $dist_{MinPts}(i) = \max \{dist(i,q) \mid q \in N^{MI}_{MinPts}(i)\}$, *and the* concept core-distance *of instance* i*, denoted by* $ConceptCoreDist^{\varepsilon}_{MinPts}(i)$*, is defined as follows:*

$$ConceptCoreDist^{\varepsilon}_{MinPts}(i) = \begin{cases} dist_{MinPts}(i) & : \; dist_{MinPts}(i) \leq \varepsilon \\ \infty & : \; dist_{MinPts}(i) > \varepsilon \end{cases}.$$

The definition of the concept core-distance guarantees that the dense areas captured in the reachability plot are based on at least *MinPts* different MI objects. To derive an instance plot based on the concept core-distance, we additionally need to adjust the definition of the reachability distance to the concept reachability distance.

Definition 9.10 (Concept reachability distance)
Let MinPts $\in \mathbb{N}$, $\varepsilon \in \mathbb{R}^+$. *The* concept reachability distance *of an instance* i

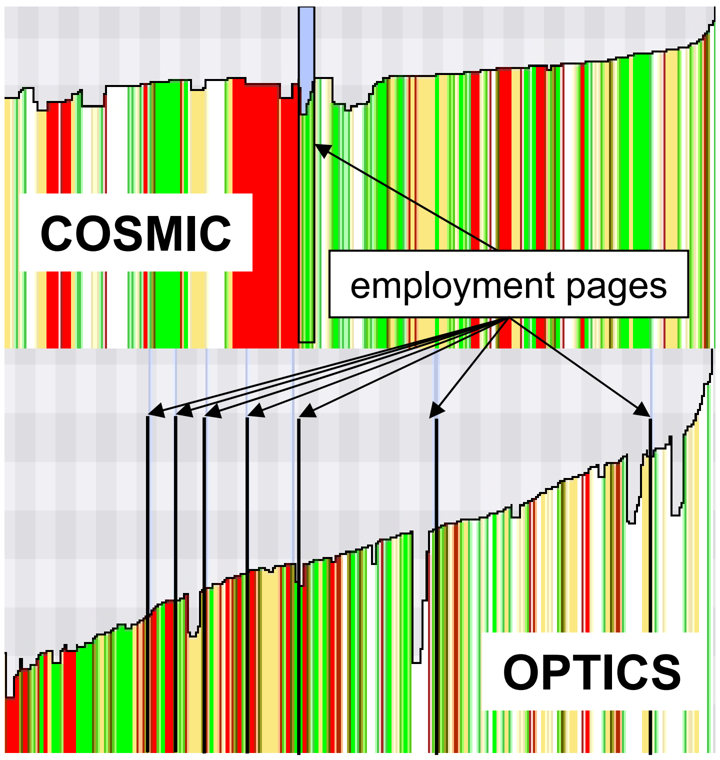


Figure 9.2: Comparison of a reachability plot created by COSMIC (upper plot) and OPTICS (lower plot) on a website dataset.

to another instance j w.r.t. ε and MinPts, denoted by $ConceptReachDist^{\varepsilon}_{MinPts}(i,j)$, is defined by

$$ConceptReachDist^{\varepsilon}_{MinPts}(i,j) = \\ \max\{ConceptCoreDist^{\varepsilon}_{MinPts}(i), dist(i,j)\}$$

Let us note that it is still possible to reach all other instances, even those from the same MI object. However, the reachability distance is at least the concept core-distance of the first instance in the predicate. The rest of the algorithm proceeds as described in [ABKS99]. To conclude, the clusters contained in the reachability plot of COSMIC generalize instances from at least *MinPts* MI objects and are thus suitable to describe clusters

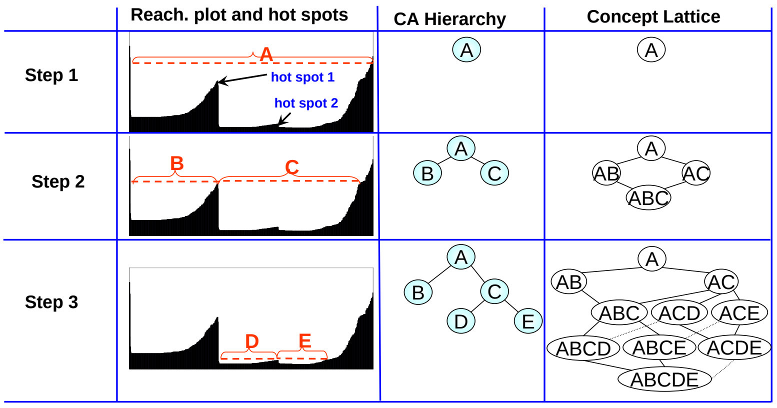


Figure 9.3: Example of derived CA hierarchy and concept lattice. The left column displays the reachability plot, the middle column the CA hierarchy and right column the maximum concept lattice that could occur.

that need to contain at least *MinSup* $\leq$ *MinPts* MI objects. Let us note that in case *MinSup* $>$ *MinPts* there might be concepts that cannot be used to describe an MI cluster. Figure 9.2 compares the reachability plot of COSMIC to the plot generated by ordinary OPTICS on a dataset of webpages taken from 46 sites. In the upper plot generated by COSMIC, it can be seen that pages concerning "employment" still cluster well even though they belong to various websites. In the lower plot generated by OPTICS the same pages do not cluster at all because the similarity of pages belonging to the same website prevents the detection of this descriptive concept.

The resulting cluster order implies a hierarchical clustering of instances. However, not all of the clusters might be useful for describing a concept and are thus suitable CAs. Therefore, it is not necessary for COSMIC to derive the complete cluster hierarchy of instances in order to find all potential CAs. Instead, COSMIC collects so-called hot spots while generating the plot itself. The hot spots mark positions in the plot where a more general instance cluster can be separated into two more specialized clusters. Technically, a hot spot is a position in the reachability plot, where the reachability distance is smaller on the right side and is smaller or equal on the left side. Thus, a hot spot corresponds to a peak or the rightmost position of a plateau in the reachability plot. We will now specify a hot spot more formally:

Definition 9.11 (Hot Spot)
Let $reach(i)$ denote the reachability distance at the ith position of a given reachability plot. Then, i is the position of a hot spot if both of the following conditions hold:

$$\begin{aligned} &(1) \quad reach(i) > reach(i+1) \\ &(2) \quad \exists l \in \mathbb{N} : reach(i-l) < reach(i) \\ &\qquad \wedge \forall k : (i-l) < k < i : reach(i) = reach(k) \end{aligned}$$

Two examples of hot spots are illustrated in Figure 9.3. To derive all hot spots from a reachability plot, it is not necessary to perform an additional scan of the reachability plot. Instead, it is possible to collect all necessary hot spots while generating the reachability plot by monitoring the local minima and maxima which were encountered so far. Figure 9.4 explains the collection of hot spots in more detail.

To subsume, the result of the first step of COSMIC is a reachability plot that guarantees that the induced clusters provide concepts which are suitable to describe MI clusters. Additionally, a set of so-called hot spots within the reachability plot is derived indicating the splitting points of the clusters in the plot.

9.4.2 Deriving Attributes and Concepts

Having analyzed the general patterns in the set of all instances, the second step of COSMIC generates all concepts that contain at least *MinSup* MI objects. *MinSup* is the only parameter that has to be specified for the actual

```
ConceptReachPlot(MIObject[] $\mathcal{DB}$, real $\varepsilon$, integer $MinPts$)
    ConceptReachabilityPlot plot;
    $I \leftarrow \bigcup_{o \in \mathcal{DB}} o$;
    plot.prevDist $\leftarrow \infty$;
    plot.prevPos $\leftarrow 0$;
    plot.isAscending $\leftarrow$ FALSE;
    plot $\leftarrow$ Cluster $I$ using $ConceptCoreDist$,
       $ConceptReachDist$ and ConceptReachabilityPlot.add;
    sort hot spots in plot descending w.r.t. $ConceptReachDist$;
    RETURN plot;

ConceptReachabilityPlot.add (Instance $i$)
    add $i$ to plot;
    IF prevDist $>$ $i.ConceptReachDist$ DO
       IF isAscending DO
          add new hotSpot with
             $ConceptReachDist \leftarrow$ prevDist;
             position $\leftarrow$ prevPos;
       END IF
       isAscending $\leftarrow$ FALSE;
       prevDist $\leftarrow$ $i.ConceptReachDist$;
       prevPos $\leftarrow$ current length of plot;
    ELSE IF prevDist $<$ $i.ConceptReachDist$ DO
       isAscending $\leftarrow$ TRUE;
       prevDist $\leftarrow$ $i.ConceptReachDist$;
       prevPos $\leftarrow$ current length of plot;
    END IF
```

Figure 9.4: Pseudocode: collection of hot spots.

extraction of MI clusters. Let us note that $MinSup = 1$ allows COSMIC to derive any concept that can be found in the dataset w.r.t. to the given reachability plot. However, since most of the concepts will be rather specific to a single MI object, the resulting concept lattice will be very complex containing a large variety on very specific concepts that are not interesting from the data mining point of view.

The input for the second part of COSMIC is the reachability plot derived in the first step and the hot spots collected within this plot. The general idea of this algorithm is to employ a top-down sweep line algorithm to the reachability plot which simultaneously extracts CAs and concepts. The algorithm starts with a trivial set of one CA, namely c_{all}, and a trivial concept which corresponds to the complete dataset $\mathcal{DB}$ and is described by $\{c_{all}\}$. Before starting the sweep line algorithm, we first of all sort the hot spots descending w.r.t. their reachability distance in the plot. The sweep line stops at each hot spot in this generated order and determines the CA the hot spot is contained in. In the following, we will refer to this CA as the split CA. If the split CA is not used for describing any concept, we already can examine the next hot spot because there cannot be any concept being described by any SubCA of the split CA. This observation can be formalized in the following monotonicity criterion:

Lemma 9.1 (monotonicity criterion)

Let $C = \{c_1, \ldots, c_m\}$ be a set of CAs over the feature space F and $M = \{M_1, \ldots, M_l\}$ be a set of concepts which are described by C. Furthermore, let $c_g \in C$ be some CA and let $M_g \in M$ be some concept with $c_g \in Desc(M_g)$. Then for any subconcept M_s that can be described by

$$Desc(M_g) \setminus c_g \bigcup SubConAttr(c_g)$$

the following rule holds:

$$|M_G| \leq k \Rightarrow |M_s| \leq k.$$

Proof.

$$Desc(M_s) = Desc(M_g) \setminus c_g \bigcup SubConAttr(c_g)$$

$$\begin{aligned} &\Rightarrow \quad M_s \subseteq M_g \\ &\Rightarrow \quad |M_g| \geq |M_s| \end{aligned}$$

$\square$

If the split CA is an element of any concept description, the algorithm determines the expansion of the new SubCAs. Therefore, the plot is traversed in both directions beginning with the hot spot and ending with a position for which the reachability distance is again at least as large as at the hot spot. Let us note that it might be necessary to cross some plateau, i.e. an area of

```
deriveConceptLattice(ReachabilityPlot plot, integer $MinSup$)
  init ConceptLattice conceptLattice;
  init CAHierarchy $h\_CA$;
  $h\_CA.rootCA \leftarrow \{i | i \in plot\}$
  IF $|\{MiObj(i) | i \in h\_CA.rootCA\}| > MinSup$ THEN
    conceptLattice.$rootConcept$ ← new Concept( $\{h\_CA.rootAttribute\}$);
    FOR EACH hotSpot FROM plot DO
      $c$ ← leaf node in $h\_CA$ containing hotSpot;
      IF $\exists concept \in$ conceptLattice: $c \in ClustDesc(MICluster)$ THEN
        ($c_1$,$c_2$) ← deriveSubcluster($c$, hotSpot);
        c.addSubConcept($c_1$,$c_2$);
        FOR EACH concept $B$ with $c \in ClustDesc(B)$ DO
          $m(c_1) \leftarrow \{O \in B | \exists i \in O : i \in c_1\}$;
          $m(c_2) \leftarrow \{O \in B | \exists j \in O : j \in c_2\}$;
          $m(c_1c_2) \leftarrow \{O \in B | \exists i, j \in O : i \in c_1 \wedge j \in c_2\}$;
          IF $(card(m(c_1)) > MinSup)$ THEN
            concept $sub_1$ = new Concept($m(c_1)$,B.desc.add($c_1$));
            B.addSubConcept($sub_1$);
          ENDIF
          IF$(card(m(c_2)) > MinSup)$ THEN
            concept $sub_2$ = new Concept($m(c_2)$,B.desc.add($c_2$));
            B.addSubConcept($sub_2$);
          ENDIF
          IF$(card(m(c_1c_2) > MinSup)$
            concept $sub_3$ = new Concept($m(c_1c_2)$,B.desc.add($c_1, c_2$));
            $sub_1$.addSubConcept($sub_3$);
            $sub_2$.addSubConcept($sub_3$);
          ENDIF
        END FOR
      END IF
    END FOR
  END IF
  RETURN conceptLattice and $h\_CA$;
```

Figure 9.5: Pseudocode: concept lattice.

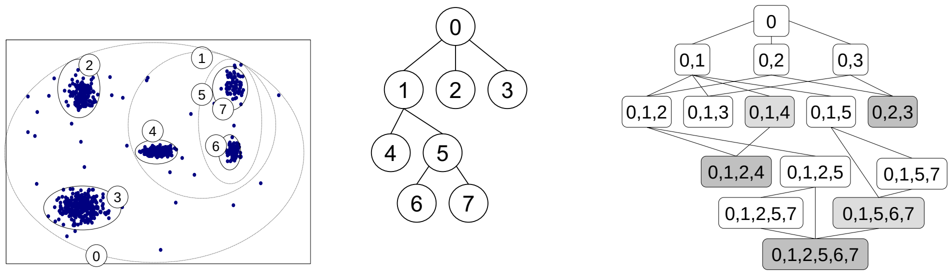
the plot having the same reachability distance, before finding the indicated cluster. The CAs are now stored in the CA hierarchy below the split CA. Since it is possible to split a CA which is an inner node of the CA hierarchy, the degree of the CA hierarchy is arbitrary.

After determining the SubCAs of the split CA, COSMIC has to check if it is possible to extend any concept that is described by the split CA. Thus, all concepts being described by the split CA are checked if they can be specialized into subconcepts that can be described by any of the new SubCAs or the combination of both. If any of the resulting cluster descriptions denotes a cluster having more than *MinSup* elements, the concept lattice is extended. In the first two cases, the new subconcept can be described by one of the SubCAs and the new concepts are direct subconcepts of the concept that is currently examined. In the third case, the existence of a concept containing both SubCAs implies that both previous cases form also a subconcept because the new concept contains the intersection of the subconcepts generated in the previous cases. Thus, if the concept being described by both SubCAs has more than *MinSup* objects, then the concepts containing only one of the SubCAs must also have at least *MinSup* members. Let us note that determining the cardinality of the subconcepts can be done quite efficiently. For every element of a concept that might be split, we simply have to check if it has at least one instance in any of the new subconcepts and then combine the results. After a concept is processed that contains the split CA in its description, additional links have to be added to the concept lattice between each pair of newly constructed concepts for which there was a subconcept relation between their father concepts.

The algorithm terminates when there are no more hot spots that could be processed. Figure 9.3 illustrates the process of CA extraction and concept expansion on a simple example having a plot based on two hot spots. The left column displays hot spots and corresponding concept attributes in the reachability plot, the middle column displays the CA hierarchy that can be derived from this plot. The right column contains all possible concept descriptions that can be derived from the CA hierarchy.

Let us note that in an ordinary task it is very unlikely that there exists a concept corresponding to any possible combination of CAs. Thus, a real concept lattice is usually smaller than that displayed in the right column of figure 9.3.

Figure 9.5 summarizes the second step of COSMIC in pseudocode notation.



(a) data space **(b)** CA Hierarchy **(c)** Concept Lattice

Figure 9.6: COSMIC on artificial data.

9.5 Experimental Evaluation

In this section, we present the results of our experimental evaluation w.r.t. effectiveness, efficiency and parameter insensitivity. All experiments described below were carried out on a workstation equipped with 2 Opteron 1.8 GHz processors and 8 GB memory. The compared algorithms are implemented in Java 1.5.

9.5.1 Experiments on Synthetic Datasets

To illustrate the capability of COSMIC to identify a specific concept lattice, we generated an artificial 2D dataset of 100 MI objects. Each instance belonging to some MI object belongs to 1 of 5 CAs, or noise, respectively. To generate MI objects, we assumed 3 basis types, each specified by a subset of the CAs. Furthermore, we introduced noise as an additional CA possibly occuring in all of the basis types. An MI object belonging to one of the types is built by selecting ten times a random CA describing the type. To generate an instance, we used a fixed Gaussian for each CA. The Gaussian modeling noise exhibits a relatively large variance. Figure 9.6 displays the results of COSMIC with *MinPts* = 20 and *MinSup* = 10. The left column depicts the instances in the 2D dataset. The middle column shows the correspond-

Table 9.1: Description of the test datasets.

	Dataset DS 1	**Dataset DS 2**	**Dataset DS 3**	**Dataset DS 4**	**Dataset DS 5**
Name	MUSK 1	MUSK 2	Dobson & Doig	Brenda	Web
No. MI-Obj.	92	102	969	10,254	46
Avg. No. of Inst. per MI-Obj.	5.2	64.7	2.4	1.977	7.72
No. of Classes	2	2	7	115	4

ing CA hierarchy containing the 5 clusters as leaf concept attributes. The right column illustrates the concept lattice derived by COSMIC. The grey nodes are the concepts which are the most specialized for some MI object. The dataset was generated employing the following basis types $(2, 3)$, $(2, 4)$ and $(2, 6, 7)$. In our concept lattice, these concepts were found as $(0, 2, 3)$, $(0, 1, 2, 4)$ and $(0, 1, 2, 5, 6, 7)$ w.r.t. to the given CA hierarchy. Additionally, some of the objects were assigned to the more general concepts $(0, 1, 4)$ and $(0, 1, 5, 6, 7)$. Since not all MI objects must contain a representative instance for each CA describing its basis type, these MI object could not be assigned to the concept corresponding to their basis type. However, the MI objects were still assigned to generalizations of these concepts. To conclude, COSMIC derived all valid concepts that could be found in the given dataset. Let us note that the concept lattice is complete though not all generalizations of any concept can be found in it. For example, there is no concept $(0, 1, 2, 5, 6)$ because the MI objects belonging to this concept are exactly the same as for the concept $(0, 1, 2, 5, 6, 7)$. Therefore, $(0, 1, 2, 5, 6)$ is technically not a concept w.r.t. Definition 9.6. This is indicated in the lattice by an direct edge between the concept $(0, 1, 2, 5)$ and $(0, 1, 2, 5, 6, 7)$.

9.5.2 Experiments on Real-World Datasets

Datasets. Table 9.1 provides a summary of the 5 real-world datasets used in our evaluation. The MUSK 1 and MUSK 2 datasets were taken from the UCI repository [NHBM98] and describe a set of molecules. The MI objects in MUSK 1 (DS 1) and MUSK 2 (DS 2) are labeled by human experts. The relevant class represents all molecules having a musky smell. The feature vectors of the MUSK datasets have 166 numerical attributes describing the molecules w.r.t. their exact shapes and their spatial conformations. Let us note that this is the classic benchmark dataset for multi-instance learning as proposed in [DLLP97b]. The Dobson&Doig (DS 3) and BRENDA (DS 4) datasets consist of high-resolution data of enzymes taken from the protein data bank (PDB)[1]. Each enzyme comprises several chains given by amino acid sequences. In order to derive feature vectors for these instances, we derived a histogram over the occurrences of the 20 amino acids. Additionally, we aggregated the amino acids to 6 different exchange groups, considering alltogether 26 dimensions. The class labels of the Dobson&Doig (DS 3) dataset were obtained as described in [DD05]. The class labels for the enzymes in DS 4 correspond to the level three of the enzyme class numbers of the comprehensive enzyme information system BRENDA[2]. The last dataset (DS 5) contains 355 webpages taken from WebKB[3]. The webpages belong to 46 websites corresponding to 4 classes in the health care sector. We used 8,000 dimensional feature vectors reflecting the occurrence of certain words in webpages.

Effectivity. In order to demonstrate the advantages of COSMIC analyzing MI datasets in an unsupervized way, we compared COSMIC to density-based and k-medoid clustering algorithms working on set-valued distance functions. Therefore, we compared the effectiveness and the efficiency of

[1] http://www.rcsb.org/pdb/
[2] http://www.brenda.uni-koeln.de/
[3] http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-11/www/wwkb/

COSMIC with that of PAM and OPTICS. To enable PAM and OPTICS to compare MI objects, we used the Hausdorff distance (HD) [EM97], the minimum Hausdorff distance (mHD) [WZ00] and the Sum of Minimum Distances (SMD) [EM97].

COSMIC provides us with a concept lattice that cannot be directly compared to the flat class labels in our test datasets. In order to find a unique mapping of MI objects to one dedicated cluster, we determine the most specialized concept containing the MI object. The most specialized concept is always the concept having the most CAs in its description. If this method does not provide a unique mapping, we additionally weight each CA with the inverse number of instances supporting the CA and calculate the sum over all CAs in the concept description. Thus, a description of very specialized CAs is favored. As a result, each MI object is labeled with a single concept and the set of all used concepts provides a flat clustering of MI objects. To assess the quality of these clusters, we determined the majority class in each cluster w.r.t. to the provided class labels and calculated the relative number of objects belonging to the majority class, i.e. the precision. The precision now indicates whether the objects in the cluster are indeed similar or not. To combine the precision over each cluster, we computed the weighted sum over all clusters using the size of each cluster as its weight.

OPTICS provides us with a reachability plot w.r.t. the mentioned distance functions. Afterwards, we calculated the cluster hierarchy by applying a cluster extraction method that is based on hot spots. On the resulting clustering, we mapped each object to its most specialized cluster and summed up the precision of the most special cluster an MI object was placed in. For the clustering computed by PAM, the precision was determined directly. We compared all results of the compared clustering approaches w.r.t. an equal number of clusters. On the average, the derived number of clusters was about four times the number of classes. Since the classes are usually rather complex and do not really indicate clusters, we argue that it is feasible to allow one

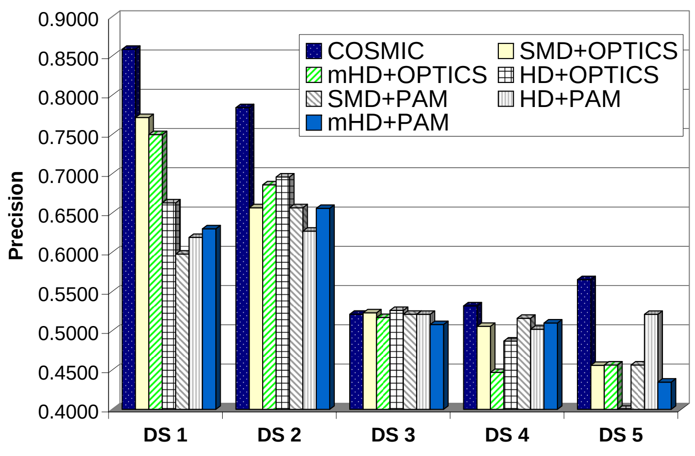


Figure 9.7: Average Cluster Precision.

class to be represented by more than one cluster. However, in some cases PAM generated empty clusters, i.e. PAM found less clusters than indicated by the parametrization. Furthermore, in some cases OPTICS did not derive an adequate number of clusters regardless of any parameter setting. In these cases, we compared the results of COSMIC to the best results of PAM and OPTICS, respectively.

The results are illustrated in Figure 9.7. For all datasets, COSMIC achieved a higher precision than the other methods. This suggests that the MI clusters derived from the concept lattice are more precise than using established set-valued distance functions. For example, for DS 1 COSMIC achieved a precision of 0.858, whereas the best result of the remaining methods was 0.772. The second best clustering was calculated by OPTICS using Hausdorff distance.

To conclude, COSMIC showed a superior precision compared to its comparison partners.

To illustrate the understandability of the learned cluster descriptions, we will describe some concepts found in the concept lattice of the website dataset DS 5. The cluster hierarchy displayed several leaf concepts that

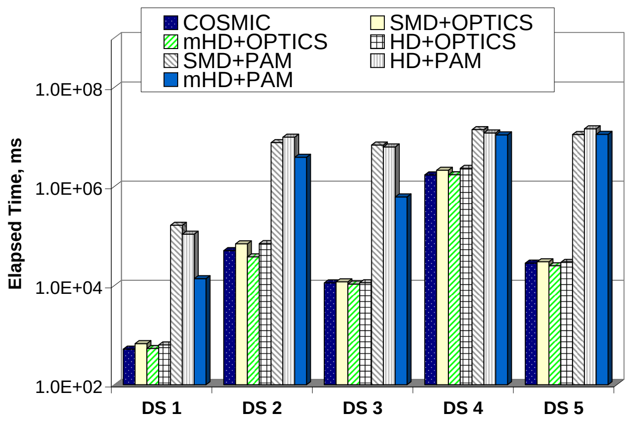


Figure 9.8: Complete Runtime of COSMIC and its comparison Partners.

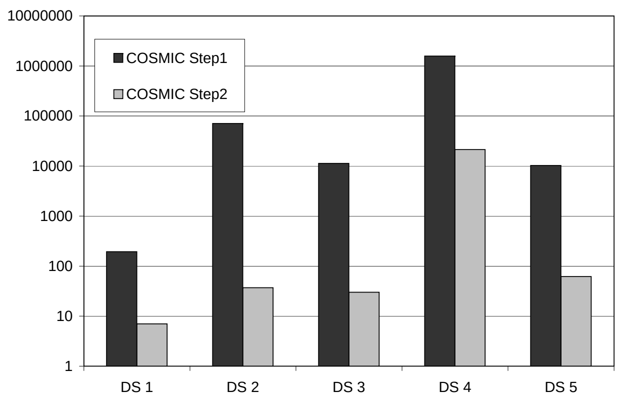


Figure 9.9: Runtime comparison between both steps of COSMIC.

were easy to understand. Among them were the following concepts: *Menu Pages*, *Contact Pages*, *Employment Pages*, *Quarter Results*, *Disclaimers* and *Company Descriptions*.

We identified concepts of websites that were described by *Employment Pages*, *Company Descriptions* and *Contact Pages*. The member websites of this concept represent companies from the biotech area that were trying to recruit new employees. Another concept corresponding to companies in the major drug industry was described by the concepts: *Menu Pages*, *Contact Pages* and *Disclaimers*.

Efficiency. To measure the efficiency of COSMIC and its competitors, we compared the elapsed runtime. For COSMIC, we consider the time spent for deriving the reachability plot, the CA hierarchy, and the concept lattice. For OPTICS, the runtime consists of the time for deriving the reachability plot and the cluster hierarchy. The runtime of PAM can be measured directly. The results are depicted in Figure 9.8. COSMIC showed a runtime behavior which is comparable to the best of the remaining methods. For example, on DS 4, the largest of the used datasets, COSMIC needed 1714 seconds, whereas OPTICS with minimum Hausdorff distance needed 1724 seconds, OPTICS with SMD ran in 2124 seconds, and OPTICS with Hausdorff distance ran in 2324 seconds. Let us note that the use of CLARANS [NH94] would have been more efficient, but less effective than PAM. However, since PAM already shows inferior effectiveness (cf. Figure 9.7) compared to COSMIC, using CLARANS instead of PAM seemed inappropriate.

Another interesting result can be observed when comparing the runtimes of the two steps of COSMIC. As indicated in Figure 9.9, the time that was spent on deriving the reachability plot from the set of all instances took on the average about two orders of magnitude more time than the second step deriving the concept lattice from the plot. Let us note that we had to use a logarithmic scale to plot the bar diagram because of the huge differences between the values. This is an important observation because the worst case complexity of the second step is exponential in the number of derived CAs while the complexity of the first step is only quadratic w.r.t. to the number of instances in all MI objects. However, in our experiments it turned out that the number of CAs that can be used to describe a concept is rather low. Thus, deriving the concept lattice required only an disappearingly small fraction of the complete runtime in all of the experiments we performed. Therefore, COSMIC displayes a runtime behavior that is comparable to running OPTICS on the set of all instances.

Insensitivity to parameter setting. We analyzed the behavior of

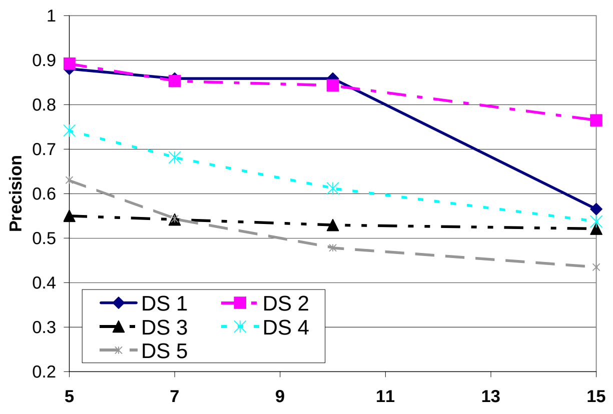


Figure 9.10: Insensitivity w.r.t. *MinPts*.

COSMIC w.r.t. different parameter settings. As mentioned above, the parameter ε is only influencing the efficiency if an index structure is available for managing the data. Thus, we generally used $\varepsilon = \infty$. Therefore, we performed several runs of COSMIC on our test bed with varying parameters *MinPts* and *MinSup*, respectively. The experimental results are illustrated in Figure 9.10 and Figure 9.11. The precision of COSMIC on all datasets stays relatively stable with varying *MinPts* and *MinSup*. The variation of precision is strongest for the DS 1 ranging from 0.66 to 0.85 for varying *MinPts*, and ranging from 0.68 to 0.83 for varying *MinSup*. This effect can be explained by the fact that the difference of 5 in the parameter *MinPts* or *MinSup*, respectively, corresponds to approximately 5.5 percent of the dataset containing only 92 MI objects. As a result it is not necessary to spend a large amount of time optimizing the parameter settings.

9.6 Conclusions

In this chapter, COSMIC was proposed, a method for deriving concept lattices from MI datasets. An MI object is specified by a set of feature representations that belong to one and the same data space. COSMIC describes

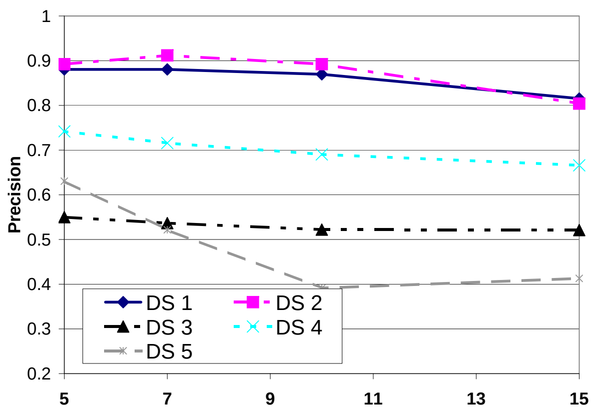


Figure 9.11: Insensitivity w.r.t. *MinSup*.

concepts of MI objects by sets of so-called cluster attributes (CAs). A CA is a common pattern in the data space of instances that might be used to characterize at least *MinPts* MI objects. To use CAs on different abstraction levels and thus, to be less dependent on the parameter setting, COSMIC employs a hierarchy of CAs. The CA hierarchy is calculated employing density-based hierarchical clustering while considering that a CA has to describe instances from at least *MinPts* MI objects. The second step of COSMIC extracts a concept lattice along with the CA hierarchy used for the concept descriptions. In our experimental evaluation, we compare COSMIC to two distance-based approaches for clustering MI data on 5 real-world datasets. The results demonstrate that COSMIC generates more precise clusterings w.r.t. a reference class set. Additionally, we show that COSMIC scales well to even larger datasets and is very insensitive to the choice of its two main parameters *MinPts* and *MinSup*.

Chapter 10

Density-based Clustering of Multi-Represented Objects

One of the challenging properties of advanced database systems is the fact that objects are usually described by multiple representations. Thus, the idea of clustering multi-represented objects is getting increasing attention from the research community. In this chapter, we propose a density-based solution to the subspace clustering problem. First, we motivate the proposed techniques in Section 10.1, Next, we review and discuss recent methods and related work in Section 10.2. Section 10.3 formalizes the problem and introduces a multi-represented version of DBSCAN using an intersection and an union paradigm for the combination of multiple representations. In Section 10.4, we start with a theoretical discussion of the semantics problem leading to the definition of combination trees for multiple semantics. Section 10.5 describes the multi-represented version of OPTICS which is capable of deriving a cluster hierarchy for any given combination tree containing intersection and union operators. In our experimental evaluation in Section 10.6.2, it is shown that the cluster quality can be improved using the proposed methods. Finally, we conclude the chapter in Section 10.7 with a short summary.

10.1 Introduction

In recent years, the research community spent a lot of attention to clustering resulting in a large variety of different clustering algorithms [HK06]. However, all those methods are based on one representation space, usually a vector space of features and a corresponding distance measure. But for a variety of advanced applications such as biometrics, biomolecular data, CAD- parts or multimedia data, it is problematic to find a common feature space that incorporates all given information. Molecules like proteins are characterized by an amino acid sequence, a secondary structure and a 3D representation. Additionally, protein databases such as Swissprot [BBA$^+$03] provide meaningful text descriptions of the stored proteins. In CAD-catalogues, the parts are represented by some kind of 3D model like Bezier curves, voxels or polygon meshes and additional textual information like descriptions of technical and economical key data. Another example is biometric data comprising speech patterns, fingerprints and facial features.

To cluster multi-represented data using the established clustering methods would require to restrict the analysis to a single representation or to construct a feature space comprising all representations. However, the restriction to a single feature space would not consider all available information and the construction of a combined feature space demands great care when constructing a combined distance function. In Section 10.3 of this chapter, we propose a method to integrate multiple representations directly into the clustering algorithm. Our method is based on the density-based clustering algorithm DBSCAN [EKSX96] that provides several advantages over other algorithms, especially when analyzing noisy data. Since our method employs a separated feature space for each representation, it is not necessary to design a new suitable distance measure for each new application. Additionally, the handling of objects that do not provide all possible representations is integrated naturally without defining dummy values to compensate for the missing representations. Last but not least, our method does not require a combined index structure, but benefits from each index that is provided for a single representation. Thus, it is possible to employ highly specialized index structures and filters for each representation.

Basically, we can distinguish two problems when clustering multi-represented objects, comparability and semantics. The comparability problem subsumes several issues when comparing features, distances or statements from different representations. The semantics problem is caused by differences between the knowledge that can be derived from each representation. For example, two images described by very similar text annotations are very likely to be very similar as well. On the other hand, if the words describing two images are completely disjunctive the implication that both images are dissimilar is rather weak as it is possible to describe the same object using completely different sets of words. Another type of semantics can be found in color histograms. An image of a plane in blue skies might provide the same color distribution as a sailing boat in the water. However, if two color images have completely different colors, it is usually a strong hint that the images are really dissimilar.

In Section 10.4, we will discuss how to exploit multiple representations with varying semantics. We distinguish two types of representations and show which basic combination method is used for which representation type. To combine a set of representations containing both types, we introduce so-called combination trees that can be used to describe a large variety of combination rules. To employ these combination trees for clustering, we introduce a multi-represented version of the hierarchical density-based clustering algorithm OPTICS. OPTICS derives so-called cluster orderings and is quite insensitive to the parameter selection. The introduced version of OPTICS is capable to derive meaningful cluster hierarchies with respect to an arbitrary combination tree.

10.2 Related Work

A few clustering approaches appropriate for multi-represented objects like an algorithm for spectral clustering [DS05], a version of Expectation Maximization (EM) clustering [BS04] and the framework of reinforcement clustering [WZC+03] are proposed (cf. Section 2.2.2 in Chapter 2 for details). The proposed approaches result in a partitioning clusterings of the data spaces, which makes the maximization of the agreement between local models a beneficial goal to optimize. However, in a density-based setting, there is an arbitrary number of clusters and there are no explicit clustering models that can be optimized to agree with each other. Furthermore, the three formerly mentioned approaches do not consider any semantic aspect of the underlying data spaces.

In addition, the reinforcement clustering [WZC+03] is applicable for multi-represented objects. However, due to its dependency on the data space for which the clustering is started, it is not well suited to solve the task of multi-represented clustering.

10.3 Clustering of Multi-Represented Objects with Noise

To formalize the multi-represented clustering problem, we first define the σ-neighborhood of an object o w.r.t. a representation R_i. Let $\mathcal{DB}$ be a set of objects and let $R = \{R_1, \ldots, R_m\}$ be a set of m different representations existing for objects in $\mathcal{DB}$. The *σ-neighborhood of o w.r.t. R_i* is defined as the set of objects around o with distances in representation R_i less than or equal to σ, formally:

Definition 10.1 (σ_i-neighborhood w.r.t R_i)

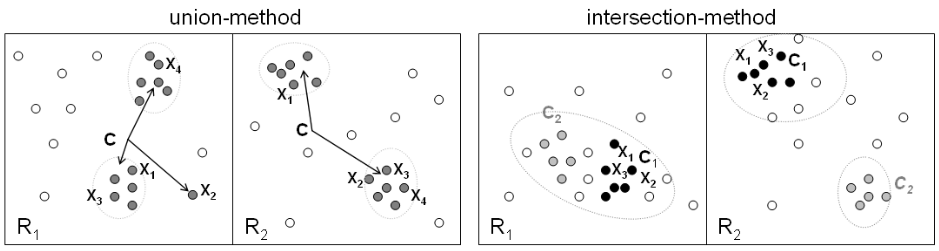


Figure 10.1: The left figure displays local clusters and a noise object that are aggregated to a multi-represented cluster **C**. The right figure illustrates, how the intersection-method divides a local clustering into clusters C_1 and C_2.

Let $o \in \mathcal{DB}$*,* $\sigma_i \in \mathbb{R}^+$*,* $R = \{R_1, \ldots, R_m\}$*, and let* d_i *be the distance function of* R_i*. The* local σ_i-neighborhood of o w.r.t. R_i*, denoted by* $\mathcal{N}_{\sigma_i}^{R_i}(o)$*, is defined by* $\mathcal{N}_{\sigma_i}^{R_i}(o) = \{x \in \mathcal{DB} \mid d_i(o, x) \leq \sigma_i\}$*.*

Note that σ_i can be chosen optimally for each representation. The simplest way of clustering multi-represented objects, is to select one representation R_i and cluster all objects according to this representation. However, this approach restricts data analysis to a limited part of the available information and does not use the remaining representations to find a meaningful clustering. Another way to handle multi-represented objects is to combine the different representations and use a combined distance function. Then any established clustering algorithm can be applied. However, it is very difficult to construct a suitable combined distance function that is able to fairly weight each representation and handle missing values. Furthermore, a combined feature space, does not profit from specialized data access structures for each representation.

The idea of our approach is to combine the information of all different representations as early as possible, i.e. during the run of the clustering algorithm, and as late as necessary, i.e. after using the different distance functions of each representation. To do so, we adapt the core object property proposed for DBSCAN. To decide whether an object is a core object, we use the local σ-neighborhoods of each representation and combine the results to a global neighborhood. Therefore, we must adapt the predicate direct density-reachability proposed for DBSCAN. In the next two subsections, we will show how we can use the concepts of union and intersection of local neighborhoods to handle multi-represented objects.

10.3.1 Union of Multiple Representations

This variant is especially useful for sparse data. In this setting, the clusterings in each single representation will provide several small clusters and a large amount of noise. Simply enlarging σ would relief the problem, but on the other hand, the separation of the clusters would suffer. The union-method assigns objects to the same cluster, if they are similar in at least one of the representations. Thus, it keeps up the separation of local clusters, but still overcomes the sparsity. If the object is placed in a dense area of at least one representation, it is still a core object regardless of how many other representations are missing. Thus, we do not need to define dummy values. The left part of figure 10.1 illustrates the basic idea.

We adapt some of the definitions of DBSCAN to capture our new notion of clusters. To decide whether an object o is a union core object, we unite all local σ_i-neighborhoods and check whether there are enough objects in the global neighborhood, i.e. whether the global neighborhood of o is dense.

Definition 10.2 (union core object)
Let $\sigma_1, \sigma_2, ..., \sigma_m \in \mathbb{R}^+$, $k \in \mathbb{N}$. An object $o \in DB$ is called union core object*, denoted by $\text{CoreU}^k_{\sigma_1,..,\sigma_m}(o)$, if the union of all local σ-neighborhoods contains at least k objects, formally:*

$$\text{CoreU}^k_{\sigma_1,..,\sigma_m}(o) \Leftrightarrow \mid \bigcup_{R_i(o)\in o} \mathcal{N}^{R_i}_{\sigma_i}(o) \mid \geq k.$$

Definition 10.3 (direct union-reachability)
Let $\sigma_1, \sigma_2, .., \sigma_m \in \mathbb{R}^+$, $k \in \mathbb{N}$. An object $p \in DB$ is directly union-reachable *from $q \in DB$ if q is a union core object and p is an element of at least one local $\mathcal{N}_{\sigma_i}^{R_i}(q)$, formally:*

$$\textsc{DirReachU}_{\sigma_1,..,\sigma_m}^{k}(q,p) \Leftrightarrow \textsc{CoreU}_{\sigma_1,..,\sigma_m}^{k}(q) \wedge \exists\, i \in \{1,..,m\} : R_i(p) \in \mathcal{N}_{\sigma_i}^{R_i}(q).$$

The predicate direct union-reachability is obviously symmetric for pairs of core objects, because the $dist_i$ are symmetric distance functions. Thus, analogously to DBSCAN reachability and connectivity can be defined.

10.3.2 Intersection of Multiple Representations

The intersection method is well suited for data containing unreliable representations, i.e. there is a representation, but it is questionable, whether it is a good description of the object. In those cases, the intersection-method requires that a cluster should contain only objects which are similar according to all representations. Thus, this method is useful, if all different representations exist, but the derived distances do not adequately mirror the intuitive notion of similarity. The intersection-method is used to increase the cluster quality by finding purer clusters.

To decide, whether an object o is an intersection core object, we examine, whether o is a core object in each involved representation. Of course, we use different σ-values for each representation to decide, whether locally there are enough objects in the σ-neighborhood. The parameter k is used to decide, whether globally there are still enough objects in the σ-neighborhood, i.e. the intersection of all local neighborhoods contains at least k objects.

Definition 10.4 (intersection core object)
Let $\sigma_1, \sigma_2, ..., \sigma_m \in \mathbb{R}^+$, $k \in \mathbb{N}$. An object $o \in DB$ is called intersection

core object, *denoted by* $\textsc{CoreIS}^k_{\sigma_1,..,\sigma_m}(o)$*, if the intersection of all its local* σ_i*-neighborhoods contain at least* k *objects, formally:*

$$\textsc{CoreIS}^k_{\sigma_1,..,\sigma_m}(o) \Leftrightarrow | \bigcap_{i=1,..,m} \mathcal{N}^{R_i}_{\sigma_i}(o) | \geq k.$$

Using this new property, we can now define direct intersection-reachability in the following way:

Definition 10.5 (direct intersection-reachability)
Let $\sigma_1, \sigma 2, ..., \sigma_m \in \mathbb{R}^+$*,* $k \in \mathbb{N}$*. An object* $p \in DB$ *is* directly intersection-reachable *from* $q \in DB$ *if* q *is an intersection core object and* p *is an element of all local* $\mathcal{N}^{R_i}_{\sigma_i}(q)$*, formally:*

$$\textsc{DirReachIS}^k_{\sigma_1,..,\sigma_m}(q,p) \Leftrightarrow \textsc{CoreIS}^k_{\sigma_1,..,\sigma_m}(q) \wedge \forall i = 1,..,m : R_i(p) \in \mathcal{N}^{R_i}_{\sigma_i}(q) \ .$$

Again, reachability and connectivity can be defined analogously to DBSCAN. The right part of figure 10.1 illustrates the effects of this method.

10.4 Handling Semantics

In the previous Section 10.3, the idea of DBSCAN has been adapted to multi-represented objects. Two different methods have been proposed to decide whether a multi-represented object is a core object: the union and the intersection method. The union method assumes an object to be a core object if at least *MinPts* objects are found within the union of its local σ-neighborhoods of each representation. The intersection method requires that at least *MinPts* objects are within the intersection of all local σ-neighborhoods of each representation of a core object. Though this method was capable to distinguish two basic semantics for combining representations, the method still suffers from two drawbacks. First, DBSCAN is quite sensitive w.r.t. to the choice of σ and second, for applications having more than two representations, the two basic combination methods often fail to achieve a good combination of representations having varying semantics.

10.4.1 A Model for Local Semantics

Since feature spaces are usually not a perfect model of the intuitive notion of similarity, a small distance in the feature space does not always indicate true object similarity. Therefore, we denote two objects that a human user would classify as similar as truly similar. To formalize the semantic problem, we can distinguish two characteristics of representation spaces:

Definition 10.6 (Precision Space)
A precision space is a data space R_i where for each data object o there exists a σ-neighborhood $\mathcal{N}_{\sigma_i}^{R_i}(o)$ in which the percentage of all truly similar data objects among data objects in $\mathcal{N}_{\sigma_i}^{R_i}(o)$ normalized to $|\mathcal{N}_{\sigma_i}^{R_i}(o)|$ exceeds a given value π. Formally, a precision space R_i is defined as:

$$\exists \sigma \in \mathbb{R}^+,\ \forall o \in \mathcal{DB} : \frac{|\mathcal{N}_{\sigma_i}^{R_i}(o) \cap sim(o)|}{|\mathcal{N}_{\sigma_i}^{R_i}(o)|} \geq \pi$$

where $sim(o)$ denotes all objects in $\mathcal{DB}$ truly similar to object o.

Definition 10.7 (Recall Space)
A recall space is a data space R_i where for each data object o there exists a σ-neighborhood $\mathcal{N}_{\sigma_i}^{R_i}(o)$ in which the percentage of all truly similar data objects among the data objects in $\mathcal{N}_{\sigma_i}^{R_i}(o)$ normalized to $|sim(o)|$ exceeds a given value ρ. Formally, a recall space R_i is defined as:

$$\exists \sigma \in \mathbb{R}^+,\ \forall o \in \mathcal{DB} : \frac{|\mathcal{N}_{\sigma_i}^{R_i}(o) \cap sim(o)|}{|sim(o)|} \geq \rho$$

where $sim(o)$ denotes all objects in $\mathcal{DB}$ truly similar to object o.

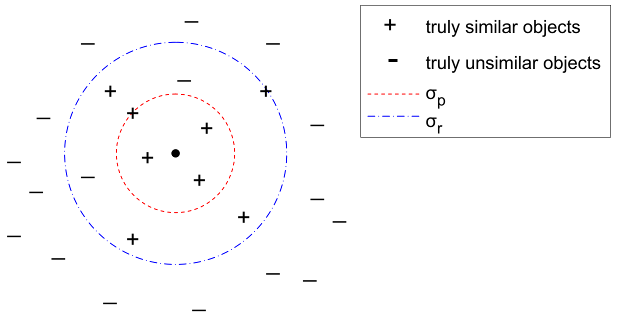


Figure 10.2: Maximal σ_p-neighborhood and minimum σ_r-neighborhood of an optimal precision and recall space.

A precision space and a recall space are called optimal, iff there exists a σ for which $\pi = 1$ and a σ for which $\rho = 1$, respectively. Figure 10.2 displays a maximal σ_p-neighborhood for object o for the case that R_i is an optimal precision space. Additionally, the figure displays the minimum σ_r-neighborhood of o for the case that R_i is an optimal recall space as well. Note that the σ_p-neighborhood is a subset of the σ_r-neighborhood in all optimal precision and recall spaces.

Though it is possible that a representation space is as good a precision space as a recall space, most real-world feature spaces are usually more suited to fulfill only one of these conditions. An example for a precision space are text vectors. Since we can assume that two very similar text annotations indicate that the described data objects are very similar as well, text annotations usually provide a good precision space. However, descriptions of two very similar objects do not have to use the same words. An object representation that is often well-suited for providing a good recall space are color histograms. If the color histograms of two color images are quite different from each other, the images are unlikely to display the same object. On the other hand, two images having similar color histograms, are not necessarily displaying the same motive.

When combining optimal precision and recall spaces for density-based clustering, our goal is to find maximum density-connected clusters where each object has only truly similar objects in its global neighborhood. In general, we can derive the following useful observations:

1. A data space that is as optimal a precision space as a recall space for the same value of σ is already optimal w.r.t our goal and thus does not need to be combined with any other representation.
2. A set of optimal precision spaces should always be combined by the union method because the union method improves the recall. If there is at least one representation R_i for all objects s that are similar to an object o, in which s belongs to the σ-neighborhood of o, the resulting combination is optimal w.r.t. recall.
3. A set of optimal recall spaces should always be combined by the intersection method because the intersection method improves the precision. If there exists no object s which is dissimilar to object o and is part of the σ-neighborhoods of o in all representations R_i, the resulting intersection is optimal w.r.t. precision.
4. Combining an optimal recall space with an optimal precision space with either union or intersection method, does not make any sense. Applying the union method is equivalent to only using the recall space and applying the intersection method is equivalent to only using the precision space.

The derived statements only hold for optimal precision and recall spaces. Since a representation is always a precision space as well as a recall space to some degree, the observations generally do not hold for the non-optimal case. For example, it might make sense to combine a very good recall space with a very good precision space, if the recall space has a good quality as a precision space as well at some other σ level. However, the implications to

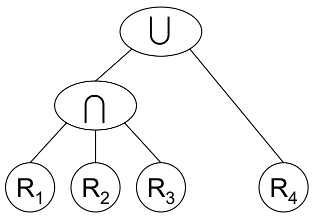


Figure 10.3: Combination tree of the image dataset.

the general case are strong enough to derive useful heuristics. A final problem for applying our model is the fact that it is not possible to determine π and ρ values for the given representations without additional information about true similarity. Thus, we have to employ domain knowledge when deriving some heuristics for building a well-suited combination of representations for clustering.

10.4.2 Combining Multiple Representations

Though we might not be able to exactly determine the parametrization for which a representation fulfills the precision and recall space conditions in a best possible way, we can still reason about the suitability of a representation for each of both conditions as in our running example of text vectors and color histograms. The most important implication of our model is that combining a good precision space and a good recall space with a rather bad precision space and a rather bad recall space, respectively will not increase the all over quality of clustering. Considering only two representations, there are only three options: use the union method for two precision spaces, the intersection method for two recall spaces or cluster only the more reliable representation in case of a mixture. For more than two representations, the combination of precision and recall spaces still can make sense. The idea is to combine these representations on different levels. Since the intersection method increases the precision and the union method increases the recall, we are able to construct recall spaces or precision spaces from a subset of the representations. To formalize this method, we will now define the so-called combination tree:

Definition 10.8 (Combination Tree)
Let $R = \{R_1, \ldots, R_m\}$. A combination tree CT for R is a tree of arbitrary degree fulfilling the following conditions:

- *$CT.root$ denotes the root of the combination tree CT.*
- *Let n be a node of CT, then $n.label$ denotes the label of n and $n.children$ denotes the children of n.*
- *The leaves are labeled with representations, i.e. for each leaf $n \in CT$: $n.label \in \{R_1, \ldots, R_m\}$.*
- *The inner nodes are labeled with either the union or the intersection operator, i.e. for each inner node $n \in CT$: $n.label \in \{\cup, \cap\}$.*

A good combination according to our heuristics can be described by a combination tree where the sons of each intersection node are all quite good recall spaces and the sons of each union node are all quite good precision spaces. Figure 10.3 displays the combination tree of the image dataset, we used in our experiments. R_1, R_2 and R_3 represent the content-based feature representations expressing texture features and color distributions. In each of these representations a small distance between the feature vectors does not necessarily indicate that the underlying images are truly similar. Therefore, we use all of these 3 representations as recall spaces. Representation R_4 consists of text annotations. As mentioned before, text annotations usually provide good precision spaces but may also provide good recall spaces. Thus, we use the text annotation as a precision space.

10.5 Hierarchical Clustering of Multi-Represented Objects

10.5.1 Normalization

In order to obtain the comparability of distances derived from different feature spaces, we perform a normalization of the distances for each representation. Let $\mathcal{DB}$ be a set of n objects and let $R := \{R_1, \ldots, R_m\}$ be a set of m different representations existing for objects in $\mathcal{DB}$.

We normalize the distance with respect to the mean value μ_i^{orig} of the original distance d_i^{orig} in representation R_i. The mean value can be calculated by sampling a small set of objects from the current representation R_i. The normalized distance between two objects $o, q \in \mathcal{DB}$ w.r.t. R_i is denoted by $\mathrm{d}_i(o, q)$ and can be calculated as follows: $\mathrm{d}_i(o, q) = \mathrm{d}_i^{orig}(o, q)/\mu_i^{orig}$.

10.5.2 Multi-Represented OPTICS

The algorithm OPTICS [ABKS99] works like an extended DBSCAN algorithm, computing the density-connected clusters w.r.t. all parameters ε_i that are smaller than a generic value of ε. Since we handle multi-represented objects, we have not only one ε-neighborhood of an object o but several ε-neighborhoods, one for each representation R_i.

In contrast to DBSCAN, OPTICS does not assign cluster memberships, but stores the order in which the objects have been processed and the information which would be used by an extended DBSCAN algorithm to assign cluster memberships. This information consists of two values for each object, its core distance and its reachability distance. To compute these information during a run of the OPTICS algorithm on multi-represented objects, we must adapt the core distance and reachability distance predicates of OPTICS to our multi-represented approach. In the following, we will show how we can use a combination tree CT for a given set of representations R to cluster multi-represented objects.

The (global) distance between two objects $o, p \in \mathcal{DB}$ w.r.t. a combination tree CT is defined as the combination of the distances of the nodes of CT.

Definition 10.9 (distance w.r.t. CT**)**
Let $o, p \in \mathcal{DB}$*,* $R = \{R_1, \ldots, R_m\}$*,* d_i *be the distance function of* R_i*,* CT *be a combination tree for* R*, and let* n *be a node in* CT*, i.e.* $n.label \in \{\cup, \cap, R_1, \ldots, R_m\}$*.*

The distance between o and p w.r.t. node $n \in CT$*, denoted by* $d^n(o,p)$*, is recursively defined by*

$$d^n(o,p) = \begin{cases} \min\limits_{c \in n.children} \{d^c(o,p)\} & \text{if } n.label = \cup \\ \max\limits_{c \in n.children} \{d^c(o,p)\} & \text{if } n.label = \cap \\ d_i(o,p) & \text{if } n.label = R_i \end{cases}$$

The distance between o and p w.r.t. CT*, denoted by* $d_{CT}(o,p)$*, is defined by*

$$d_{CT}(o,p) = d^{CT.root}(o,p)$$

The (global) ε-neighborhood of an object $o \in \mathcal{DB}$ w.r.t. a combination tree CT is defined as the combination of the ε-neighborhoods of the nodes of CT.

Definition 10.10 (ε**-neighborhood w.r.t.** CT**)**
Let $o \in \mathcal{DB}$*,* $\varepsilon \in \mathbb{R}^+$*,* $R = \{R_1, \ldots, R_m\}$*,* CT *be a combination tree for* R*, and let* n *be a node in* CT*, i.e.* $n.label \in \{\cup, \cap, R_1, \ldots, R_m\}$*.*

The ε-neighborhood of o w.r.t. node $n \in CT$*, denoted by* $\mathcal{N}_\varepsilon^o n$*, is recursively defined by*

$$\mathcal{N}_\varepsilon^o n = \begin{cases} \bigcup\limits_{c \in n.children} \mathcal{N}_\varepsilon^o c & \textit{if } n.label = \cup \\ \bigcap\limits_{c \in n.children} \mathcal{N}_{\varepsilon_i}^c(o) & \textit{if } n.label = \cap \\ \mathcal{N}_{\varepsilon_i}^{R_i}(o) & \textit{if } n.label = R_i \end{cases}$$

The ε-neighborhood of o w.r.t. CT*, denoted by* $\mathcal{N}_{CT,\varepsilon}(o)$*, is defined by*

$$\mathcal{N}_{CT,\varepsilon}(o) = \mathcal{N}_\varepsilon^o CT.root$$

Since the core distance predicate of OPTICS is based on the concept of *MinPts*-nearest neighbor (*MinPts*-NN) distances, we have to redefine the *MinPts*-nearest neighbor distance of an object o w.r.t. a combination tree CT.

Definition 10.11 (*MinPts*-NN distance w.r.t. *CT*)
Let $o \in \mathcal{DB}$*,* $MinPts \in \mathbb{N}$*,* $|\mathcal{DB}| \geq MinPts$*,* $R = \{R_1, \ldots, R_m\}$*,* CT *be a combination tree for* R*, and let* n *be a node in* CT*, i.e.* $n.label \in \{\cup, \cap, R_1, \ldots, R_m\}$*.*

The MinPts-nearest neighbors of o w.r.t. CT *is the smallest set* $NN_{CT,MinPts}(o) \subseteq \mathcal{DB}$ *that contains (at least) MinPts objects and for which the following condition holds:*

$$\forall p \in NN_{CT,MinPts}(o), \forall q \in \mathcal{DB} \setminus NN_{CT,MinPts}(o) : d_{CT}(o,p) < d_{CT}(o,q).$$

The MinPts-nearest neighbor distance of o w.r.t. CT*, denoted by* $\text{NN-DIST}_{CT,k}(o)$*, is defined as follows:*

$$\text{NN-DIST}_{CT,k}(o) = \max\{d_{CT}(o,q)\} \mid q \in NN_{CT,MinPts}(o)\}.$$

Now, we can adopt the core distance definition from OPTICS to our combination approach: If the ε-neighborhood w.r.t. CT of an object o contains at least *MinPts* objects, the core distance of o is defined as the *MinPts*-nearest neighbor distance of o. Otherwise, the core distance is infinity.

Definition 10.12 (core distance w.r.t. CT)
Let $o \in \mathcal{DB}$, $MinPts \in \mathbb{N}$, $|\mathcal{DB}| \geq MinPts$, $R = \{R_1, \ldots, R_m\}$, CT be a combination tree for R, and let n be a node in CT, i.e. $n.label \in \{\cup, \cap, R_1, \ldots, R_m\}$.

The core distance of o w.r.t. CT, ε and $MinPts$, *denoted by* $\textsc{Core}_{CT,\varepsilon,k}(o)$, *is defined by*

$$\textsc{Core}_{CT,\varepsilon,k}(o) = \begin{cases} \textsc{nn-dist}_{CT,k}(o) & \text{if } |\mathcal{N}_{CT,\varepsilon}(o)| \geq MinPts \\ \infty & \text{otherwise.} \end{cases}$$

The reachability distance of an object $p \in \mathcal{DB}$ from $o \in \mathcal{DB}$ w.r.t. CT is an asymmetric distance measure that is defined as the maximum value of the core distance of o and the distance between p and o.

Definition 10.13 (reachability distance w.r.t. CT)
Let $o, p \in \mathcal{DB}$, $MinPts \in \mathbb{N}$, $|\mathcal{DB}| \geq MinPts$, $R = \{R_1, \ldots, R_m\}$, CT be a combination tree for R, and let n be a node in CT, i.e. $n.label \in \{\cup, \cap, R_1, \ldots, R_m\}$.

The reachability distance of o to p w.r.t. CT, ε, and $MinPts$, *denoted by* $\textsc{Reach}_{CT,\varepsilon,k}(p, o)$, *is defined by*

$$\textsc{Reach}_{CT,\varepsilon,k}(p, o) = \max\{\textsc{Core}_{CT,\varepsilon,k}(p), d_{CT}(o, p)\}$$

By first normalizing the distances within the representations, we are now able to use OPTICS applying an arbitrary combination tree.

10.6 Performance Evaluation

Protein Databases. The first set of experiments with multi-represented DBSCAN and OPTICS was performed on protein data that is represented by amino-acid sequences and text descriptions. Therefore, we employed entries of the SWISSPROT protein database [BBA+03] belonging to 5 functional

Table 10.1: Description of the protein datasets.

	Set 1	**Set 2**	**Set 3**	**Set 4**	**Set 5**
Name	Isomerase	Lyase	Signal Trans-ducer	Oxidore-ductase	Transferase
Classes	16	35	39	49	62
Objects	501	1640	2208	3399	4086

groups (cf. Table 10.1) and transformed each protein into a pair of feature vectors. Each amino acid sequence was mapped into a 436 dimensional feature space. The first 400 features are 2-grams of successive amino-acids. The last 36 dimensions are 2-grams of 6 exchange groups that the single amino-acids belong to [DK02]. To compare the derived feature vectors, we employed Euclidian distance. To process text documents, we rely on projecting the documents into the feature space of relevant terms. Documents are described by a vector of term frequencies weighted by the inverse document frequency (TFIDF) [Sal89]. We chose 100 words of medium frequency as relevant terms and employed cosine distance to compare the TFIDF-vectors. Since SWISSPROT entries provide a unique mapping to the classes of Gene Ontology [BBA+03], a reference clustering for the selected proteins was available. Thus, we are able to measure a clustering of SWISSPROT entries by the degree it reproduces the class structure provided by Gene Ontology.

To have an exact measure for this degree, we employed the class entropy in each cluster. However, there are two effects that have to be considered to obtain a fair measure of a clustering with noise. First, a large cluster of a certain entropy should contribute more to the overall quality of the clustering than a rather small cluster providing the same quality. The second effect is that a clustering having a 5 % noise ratio should be ranked higher than a clustering having the same average entropy for all its clusters, but contains 50 % noise.

To consider both effects we propose the following quality measure for comparing different clusterings with respect to a reference clustering.

Definition 10.14 *Let O be the set of data objects, let $C = \{C_i | C_i \subset O\}$ be the set of clusters and let $K = \{K_i | K_i \subset O\}$ be the reference clustering of O. Then we define:*

$$Q_K(C) = \sum_{C_i \in C} \frac{|C_i|}{|O|} \cdot (1 + entropy_K(C_i))$$

where $entropy_K(C_i)$ denotes the entropy of cluster C_i with respect to K.

The idea is to weight every cluster by the percentage of the complete data objects being part of it. Thus, smaller clusters are less important than larger ones and a clustering providing an extraordinary amount of noise can contribute only the percentage of clustered objects to the quality. Let us note that we add 1 to the cluster entropies. Therefore, we measure the reference clustering K with the quality score of 1 and a worst case clustering — e.g. no clusters are found at all — with the score of 0.

To relate the quality of the clustering achieved by our multi-represented DBSCAN and OPTICS to the results of former methods, we compared the introduced density based methods to 4 alternative approaches. First, we clustered text (R_1) and sequences (R_2) separately using only one of the representations. A second approach combines the features of both representations into a common feature space (CFS) and employs the cosine distance to relate the resulting feature vectors. As the only other clustering method that is able to handle multi-represented data, we additionally compared reinforcement clustering using DBSCAN as underlying cluster algorithm. For reinforcement clustering, we ran 10 iterations and tried several values of the weighting parameter α. All approaches were run for both settings and the best results are displayed.

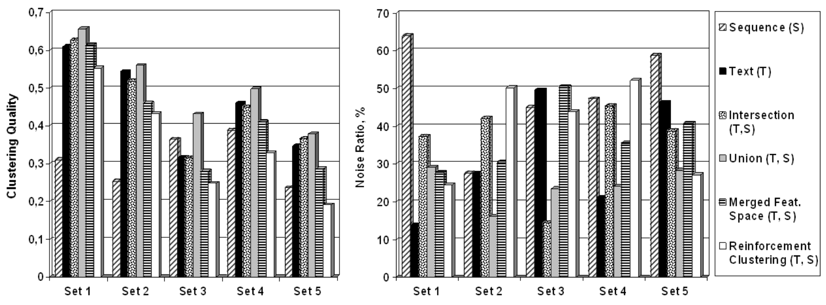


Figure 10.4: Clustering quality and noise ratio.

10.6.1 Multi-Represented DBSCAN

To demonstrate the capability of multi-represented DBSCAN, we performed a versatile experimental evaluation for protein data and image dataset. We implemented the proposed clustering algorithm in Java 1.4. All experiments were processed on a work station with a 2.6 GHz Pentium IV processor and 2 GB main memory.

Deriving Meaningful Groupings in Protein Databases. In order to apply the multi-represented version of DBSCAN, we selected the local ε-parameters as described above and we chose $k = 2$. To consider the different requirements of both intersection and union methods, for each dataset a progressive and a conservative ε-value was determined. The left diagram of figure 10.4 displays the derived quality for those 4 methods and the two variants of our method. In all five test sets, the union-method using conservative ε-values outperformed any of the other algorithms. Furthermore, the noise ratio for each dataset was between 16% and 28% (cf. figure 10.4, right), indicating that the main portion of the data objects belongs to some cluster. The intersection method using progressive ε-parameters performed comparably well, but was to restrictive to overcome the sparseness of the data as good as the union-method. **Clustering Images by Multiple Representations.** Clustering image data is a good example for the usefulness of the intersection-method. A lot of different similarity models exists for image

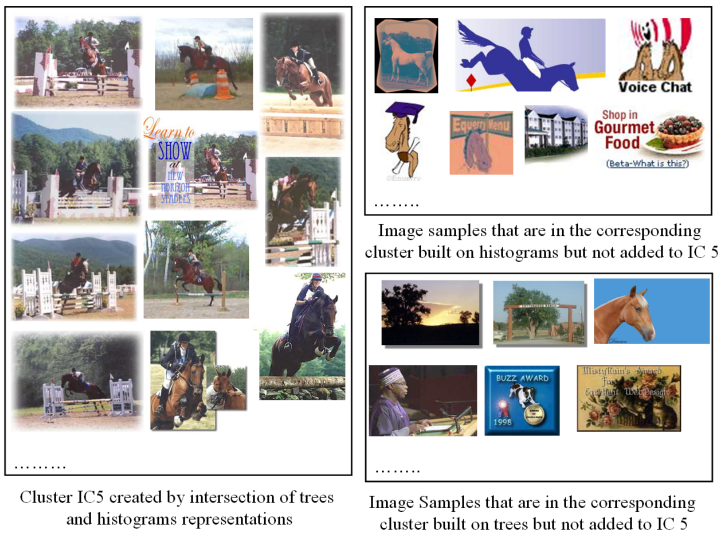

Voice Chat
Shop in Gourmet Food
(Beta-What is this?)
Equeri Menu
BUZZ AWARD
1998
Learn to SHOW
Image samples that are in the corresponding cluster built on histograms but not added to IC 5
Cluster IC5 created by intersection of trees and histograms representations
Image Samples that are in the corresponding cluster built on trees but not added to IC 5

Figure 10.5: Example of an image cluster found by multi-represented DB-SCAN.

data, each having its own advantages and disadvantages. Using for example text descriptions of images, one is able to cluster all images related to a certain topic, but these images must not look alike. Using color histograms instead, the images are clustered according to the distribution of color in the image. But as only the color information is taken into account a green meadow with some flowers and a green billiard table with some colored shots on it, can of course not be distinguished by this similarity model. On the other hand, a similarity model taking content information into account might not be able to distinguish images of different colors.

Our intersection approach is able to get the best out of all these different types of representations. Since the similarity in one representation is not really sound, the intersection-method is well-suited to find clusters of better quality for this application. For our experiments, we used two different representations. The first representation was a 64-dimensional color histogram. In this case, we used the weighted distance between those color histograms, represented as a quadratic form distance function as described for example in [HSW+95]. The second representation were segmentation trees. An image was first divided into segments of similar color by a segmentation algorithm. In a second step, a tree was created from those segments by iteratively applying a region-growing algorithm which merges neighboring segments, if their colors are alike. In [KKSS04] an efficient technique is described to compute the similarity between two such trees using filters for the complex edit-distance measure.

As we do not have any class labels to measure the quality of our clustering, we can only describe the results we achieved. In general, the clusters we got using both representations were more accurate than the clusters we got using each representation separately. Of course, the noise ratio increased for the intersection-method. We show one sample cluster of images we found with the intersection-method (see Figure 10.5). The left rectangle of Figure 10.5 contains images clustered by the intersection-method. The right rectangles display additional images that were grouped with the corresponding cluster when clustering the images with respect to a single representation. Using this method, very similar images are clustered together. When clustering each single representation, a lot of additional images were added to the corresponding cluster. As one can see, using the intersection-method only the most similar images of both representations still belong to the cluster.

10.6.2 Multi-Represented OPTICS

In order to show the capability of the multi-represented OPTICS, we performed a thorough experimental evaluation for two types of applications. We implemented the proposed clustering algorithm in Java 1.5. All experiments were performed on a work station with two 1.8 GHz Opteron processors and 8 GB main memory.

Clustering Protein Data. The first set of experiments was performed on protein data that is described by two representations R_1 and R_2: R_1 consists

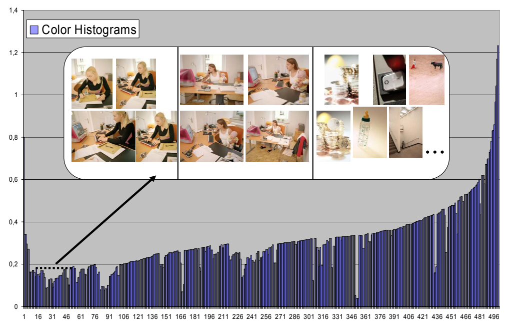
Table 10.2: Comparison of multi-represented OPTICS to different clustering approaches w.r.t. quality measure.

	Set 1	**Set 2**	**Set 3**	**Set 4**	**Set 5**
$R_1 \cup R_2$	**0.66**	**0.56**	**0.43**	**0.50**	**0.38**
R_1	0.61	0.54	0.32	0.46	0.35
R_2	0.31	0.25	0.36	0.39	0.24
CFS	0.62	0.46	0.28	0.41	0.29
RCL	0.55	0.43	0.25	0.33	0.19

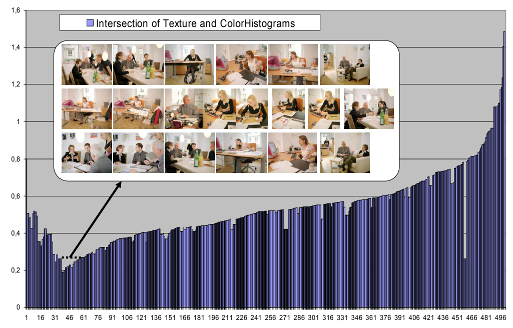
of text descriptions and R_2 stores amino-acid sequences. To evaluate the derived cluster structure C we extracted flat clusters from OPTICS plots and applied the quality measure 10.14 for comparing different clusterings w.r.t. a reference clustering K.

We applied our method to the union of representation R_1 and R_2 and compared the result to four other approaches. All approaches were run for settings described above and the best results are displayed. Table 10.2 illustrates the derived quality for our method and the four competitive methods mentioned above. As it can be seen, our method clearly outperforms any of the other algorithms.

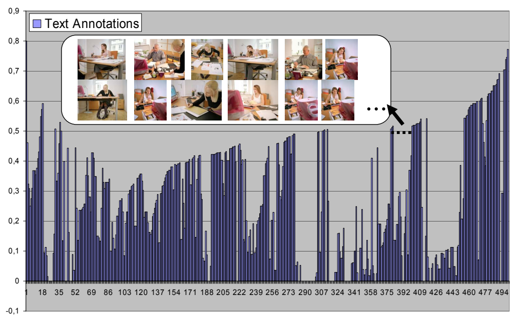
Clustering Images. We used a dataset containing 500 images manually annotated by a short text. From each image, we extracted 3 representations R_1, R_2, R_3, namely a color histogram (R_1) and two textural feature vectors (R_2, R_3). We used the HSV color space and calculated 32 dimensional color histograms based on 8 ranges of hue and 4 ranges of saturation. The textural features were generated from 16 gray-scale conversions of the images. We computed contrast and inverse difference moment using the co-occurrence matrix [HSD73]. The fourth representation R_4 consists of text annotations. For comparing text annotations, we applied the cosine coefficient and used the Euclidian distance in the rest of the representations. To verify the re-



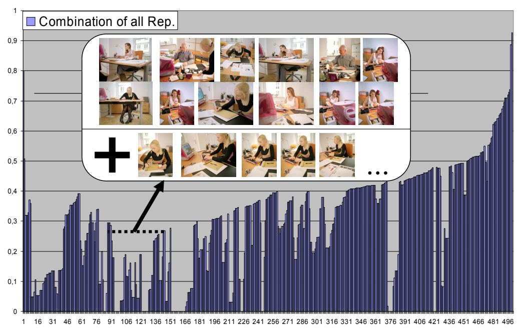
(a) OPTICS plot using only color histograms. Additionally, a representative sample set for one of the clusters is shown.



(b) OPTICS plot when employing the intersection of color histograms and both texture representations. The displayed cluster shows promising precision.



(c) OPTICS plot using only Text annotations. The displayed cluster has a high precision but is incomplete.



(d) OPTICS plot of the combination of all representations. The precise cluster observed in the text representation is completes with similar images.

Figure 10.6: OPTICS plots of image data.

sults of the found clustering, we visually verified the similarity of images in each cluster. To demonstrate the results of multi-represented OPTICS with the combination method described above, we ran OPTICS on each single representation. Additionally, we examined the clustering for the combination of color histograms and texture features using the intersection method like proposed in the combination tree. Finally, we ran OPTICS using the complete combination of image and text features. For all clusterings, we used $MinPts = 3$ and $\varepsilon = 10$. Normalization was achieved using the average distances between two objects in the dataset.

The result for the text annotations provided a very precise clustering. However, due to the fact that some annotations used different languages for describing the image, some of the clusters were incomplete. Figure 10.6(c) displays the result of clustering the text annotations. The observed cluster displays only similar objects. The cluster order derived for color histograms found some clusters. However, though the images within the clusters had similar colors, the objects were not necessarily similar. Figure 10.6(a) displays the cluster order using color histograms. It displays an image cluster containing two similar groups of images and some noise. Let use mention that the clustering of the two texture representations performed similarly. However, due to space limitations, we do not display the corresponding plots. In Figure 10.6(b) the clustering of all 3 image feature spaces using the intersection method is displayed. Though the number of clusters decreased, the quality of the remaining clusters increased considerably, as expected. The cluster shown in figure 10.6(b) contains exclusively very similar images. Finally, figure 10.6(d) displays the result on the combination of all representations. Obviously, the cluster observed for text annotations displayed in figure 10.6(c) is completed with additional similar images that are described in German instead of English language. To conclude, examining the complete clustering, the all over quality of clustering was improved by using all 4 representations.

10.7 Conclusions

In this chapter, we discussed the problem of clustering multi-represented objects. Contrary to existing approaches our proposed methods are capable of clustering cluster this kind of data using all available representations. The idea of our approach is to combine the information of all different representations as early as possible and as late as necessary. To do so, we adapted the core object property proposed for DBSCAN. To decide whether an object is a core object, we use the local ε-neighborhoods of each representation and combine the results to a global neighborhood. Based on this idea, we proposed two different methods for varying applications. For sparse data, we introduced the union-method that assumes that an object is a core object, if k objects are found within the union of its local ε-neighborhoods. Respectively, we defined the intersection-method for data where each local representation yields rather big and unspecific clusters. Therefore, the intersection-method requires that at least k objects are within the intersection of all local ε-neighborhoods of a core object.

In this chapter, we discussed also the problem of hierarchical density-based clustering of multi-represented objects having arbitrary semantics. Since each representation might have a different meaning, we first of all divided representation spaces into two basic types, precision and recall spaces. After drawing elementary conclusions about the basic types and how they should be combined using union and intersection operators, we introduced combination trees for describing arbitrary combinations of multiple representations. To cluster multi-represented objects w.r.t. a combination tree, we adapted the hierarchical clustering algorithm OPTICS to the multi-represented setting. In our experimental evaluation, we demonstrated the improvement of clustering results for an image dataset that is described by 4 representations as well as for protein datasets.

Chapter 11

Multi-Represented kNN-Classification

This chapter starts with Section 11.1 which gives an introduction to the classification of multi-represented objects. In Section 11.2, we discuss related work on speeding up kNN classification and the classification of multi-represented objects. Section 11.3 describes the use of density-based clustering to reduce the number of training instances without losing essential concepts. Additionally, our new approach to combine multi-represented classification is introduced. Section 11.4 provides an experimental evaluation based on protein data that consists of sequential and text representations. The last section sums up the introduced solutions.

11.1 Introduction

One of the most important tasks of data mining is classification. Classification learns a function $Cl : O \rightarrow C$ that maps each object $o \in O$ to the class $c \in C$ that it most likely belongs to. The class set C is a predefined set of categories. In order to make a class prediction, a classifier has to be trained. For the classification of complex objects, there are various important applications, e.g. the classification of proteins into functional catalogues or secure personal identification using several biometric characteristics. These applications yield interesting challenges to novel classification techniques.

First of all, the more complex a data object is, the more feature transformations exist that can be used to map the object to a representation suitable for data mining. Furthermore, many objects are describable by different aspects, e.g. proteins can be described by text annotations and amino acid sequences. This yields a problem for data mining in general because it is not clear which of these aspects is most suited to fulfill the given task. Therefore, it would be beneficial if a classification algorithm could employ all of the given representations of an object to make accurate class predictions. Another important aspect is that many classification algorithms rely on an object representation providing feature vectors. However, complex objects are often represented in a better way by treating them as sequences, trees or graphs. Last but not least, the number of classes in the given example applications can be exceptionally high. Gene Ontology [Con00], one of the most established class systems for proteins, currently has more then 14,000 classes and biometric databases will have to identify one special person among thousands of people. Though this problem is not directly connected to the complexity of the given data objects, it often co-occurs in the same application and should therefore be considered when selecting the classification method.

To cope with these challenges, we introduce a new classification technique based one k nearest neighbor (kNN) classification [CH67]. A kNN classifier decides the class of an object by analyzing its k nearest neighbors within the training objects. kNN classifiers are well-suited to solve the given problem because they do not have to spend additional effort for distinguishing additional classes. The new training objects are simply added to the training database and are only considered for classification if they are among the nearest neighbors of the object to be classified. Additionally, kNN classifiers can be applied to any type of object representation as long as a distance measure is available. Unfortunately, kNN classification has a major drawback as well. The efficiency of classification is rapidly decreasing with the number of training objects. Though the use of index structures such as the M-tree [CPZ97] or the IQ-Tree [BBJ$^+$00] might help to reduce query times in some cases, it does not provide a general solution. Another approach to limit the problem is the reduction of the training objects to some basic examples as proposed in [BM99]. However, these approaches are aimed at limited training data and are therefore very inefficient when applied to large training sets.

Thus, to apply kNN classification to the described classification scenario, we introduce a more efficient method to speed up kNN classification by employing density-based clustering to reduce the necessary training instances. Afterwards, we introduce a new method for the classification of multi-represented (MR) objects. The idea of the method is to determine the k nearest neighbors in a database for each representation. Then, the class prediction is derived by considering the normalized distances within each result. To demonstrate the good performance, we apply our new method to four scenarios of protein classification. Each protein is represented by an amino acid sequence and a text annotation. Our results demonstrate that density-based clustering outperforms other methods of reducing the training set for kNN classification. Furthermore, the achieved results indicate that our new decision rule for multi-represented kNN classification yields better accuracy than other classification methods applicable to large class sets.

11.2 Related Work

k Nearest Neighbor Classifier. The k nearest neighbor (kNN) classification [CH67] mentioned above classifies a new data object o by finding its k nearest

neighbors with respect to a suitable distance function. In its basic form, kNN classification predicts the class that provides the most training objects within the k-nearest neighbors. To the best of our knowledge, there exists no form of kNN classification that is directly applicable to multi-represented data objects. The common approach to apply kNN classification to this kind of data is to build a joint distance measure on the complete MR object. However, we argue that this method is not suitable to derive good results because it is not capable to weight the different representations on the basis of the given object.

Reduction of Training Instances. In the last decades, the research community introduced several methods for reduction of training instances [BM99, Gat72, RWLI75, WM97, Aha92]. All approaches try to reduce the number of instances in the training set in a way that the classifier provides comparable or even better accuracy and demands less processing time. In [WM97] the authors discuss several reduction techniques and [WM00] illustrates an experimental evaluation of these algorithms on 31 datasets. This evaluation demonstrates that the RT3 algorithm [WM97] outperforms other techniques of instance reduction for many datasets. Another approach to instance reduction is called iterative case filtering (ICF)[BM99]. This novel and effective approach to data reduction employs two steps. The first step performs so-called "Wilson editing". It detects all instances that are classified incorrectly by the kNN classifier. These instances are afterwards removed. The second step calculates for each remaining object the so-called *reachability* and *coverage* [BM99]. Every object o with $|reachable(o)| < |coverage(o)|$ is removed. The second step is iterated until no removable object exists. A broad experimental evaluation [BM02] on 30 databases compares ICF with the reduction technique RT3 [WM97]. Both algorithms achieve the highest degree of instance reduction while maintaining classification accuracy.

GDBSCAN. GDBSCAN [SEKX98] is a density-based clustering algorithm. Clusters are considered as dense areas that are separated by sparse areas. Based on two input parameters (ε and $MINPTS$), GDBSCAN defines dense regions by means of core objects. An object $o \in DB$ is called *core object*, if its ε-neighborhood contains at least $MINPTS$ objects. Usually clusters contain several core objects located inside a cluster and border objects located at the border of the cluster. In addition, the objects within a cluster must be "density-connected". GDBSCAN is able to detect clusters by one single pass over the data. The algorithm uses the fact, that a density-connected cluster can be detected by finding one of its core-objects o and computing all objects which are density-reachable from o. To determine the input parameters, a simple and effective method is described in [EKSX96]. This method can be generalized and used for GDBSCAN as well.

Classifier Fusion. The task of learning from objects, when more than a single classifier has been trained, has recently drawn some attention in the pattern recognition community as discussed in Section 2.2.2 of Chapter 2. The most important difference of co-learning approaches to our new approach of multi-represented classification is that we do not consider a semi-supervised setting. Additionally, co-training retrains its classifiers within several iterations whereas the classifiers in our approach are only trained once. In contrast to our method the hyper kernel learners optimize the use of several kernels that can be based on multiple representations within one complex optimization problem which is usually quite difficult to solve.

11.3 *k*NN-Classification of Complex Objects

In the following, we present a brief problem description. Afterwards, we introduce an approach to reduce the given training data with the help of density-based clustering. Finally, we use multiple object representations to derive accurate class predictions.

11.3.1 Problem Definition

In our given application scenario, we want to find a classifier $Cl : O \rightarrow C$ that maps each data object $o \in O$ to its correct class $c \in C$. The data space O is given by the cartesian product of m representations $R_1 \times \ldots \times R_m$. Each representation R_i consists of a feature space $F_i \cup \{-\}$. A feature space F_i may consist of varying data types. For comparing two objects $u, v \in F_i$, there exists a distance measure $dist_i : F_i \times F_i \rightarrow \mathbb{R}_0^+$. To apply our method, it is necessary that $dist_i$ is symmetric and reflexive. The symbol $\{-\}$ denotes that a particular object representation is missing. However, for a usable class prediction a tuple should provide at least one instance $r_i \in F_i$. To conclude, the task of multi-represented classification is to find a function $Cl_{mr} : (R_1 \times \ldots \times R_m) \rightarrow C$ that maps as many objects o to their correct class $c \in C$ as possible. For training, a set T of tuples (o, c) of objects $o = (r_1, \ldots, r_m)$ and their correct classes c are given to the classifier, the so-called training set. We denote in further sections the correct class of an object o by $c(o)$ and the class detected by multi-represented classification as $Cl_{mr}(o)$.

11.3.2 Density-based Training Instance Reduction

The performance of kNN classification depends on the number of objects in the training set. Though a lot of methods that reduce the training data for kNN classification have been proposed so far, most of these techniques perform poorly for large amounts of training data. In order to reduce the number of available training objects more efficiently, we suggest a novel approach – density-based instance reduction (DBIR).

The DBIR-algorithm works as follows. For each representation and each class, the training data is clustered by using the algorithm GDBSCAN. Let us note that the input parameters can be chosen as described in [EKSX96].

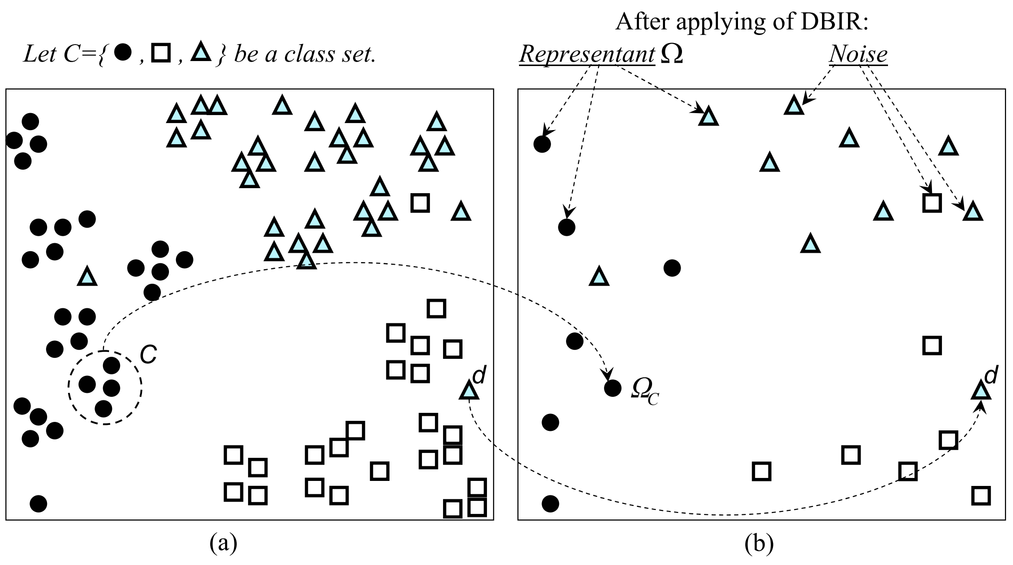


Figure 11.1: Density-based instance reduction: (a) Objects before reduction, (b) Objects after reduction.

GDBSCAN provides a set of clusters $Clust = \{Clust_1, \ldots, Clust_j, \ldots, Clust_l\}$, where $j = 1, \ldots, l$ is the index of the cluster, and additionally a set of objects N that are noise, i.e. objects that cannot be associated with any clusters. An important characteristic of GDBSCAN for our problem is that the number of found clusters l is not predefined, but a result of the clustering algorithm. Thus, the number of important concepts is determined by the algorithm and not manually. Another important advantage of GDBSCAN is that it is capable to cluster any data type as long as there is a reflexive and symmetric distance measure to compare the objects. After clustering, DBIR iterates through the set $Clust$ and determines for each cluster $Clust_j$ a representant Ω_j. The representant Ω_j is the centroid of the cluster $Clust_j$ in the case of a representation given by a vector space and the medoid of the cluster $Clust_j$ otherwise. Afterwards, all objects belonging to the set $Clust_j \setminus \Omega_j$ are removed from the dataset.

Like most other instance reduction methods, we assume that the training data for each class contains all important examples to specify a given class.

To reduce the number of training objects without losing accuracy, we have to discard the training objects that are likely to represent a concept that is not typical for the given class. Furthermore, if a typical concept is described by several training objects, we reduce the representatives of this concept to a single one to save classification time. We argue that a density-based clustering of the training objects for a given class is sufficient to decide both cases. Objects that are not typical for a given class do not have any close neighbors and are usually separated from the rest of the training set. Thus, the noise objects in a density-based clustering are likely to correspond to these objects. Of course, it is possible that a noise object alone is an important concept. However, a single object is not likely to change the decision of a kNN classifier and the decision would most likely be wrong even without the deletion. Important concepts that are represented by several training objects are usually located very closely to each other in the feature space. Thus, these concepts are likely to correspond to a density-connected cluster in our density-based clustering. For each of these clusters it is sufficient that the training set contains a single object to represent it. Figure 11.1 displays both effects in a two dimensional example. As depicted in Figure 11.1, the density-based cluster C can be reduced to a representant Ω_C. The noise object d is not removed. However, it can not change the decision of a kNN classifier with $k > 2$.

Our method has a runtime complexity of $O(\sum_{c_j \in C} |\{o \in O \mid c(o) = c_j\}|^2)$ for the case that it is not supported by index structures. ICF has a runtime complexity of $O(2 \times (\#Iteration) \times |DB|^2)$ where $\#Iteration$ is the number of iterations (in our experiments it was between 9 and 12) and $|DB|$ is the size of the database. Thus, our method is considerably faster than other state of the art feature reduction techniques.

As described above, we apply the DBIR-algorithm separately to the training objects in one representation and for one class. Afterwards we integrate all instances of a representation i into one training database DB_i. Let us note that it is possible to speed up k nearest neighbor queries in each of these training databases as long as there are suitable index structures for the given object type. For example, if the distance function is metric it might be beneficial to further increase the classification time by employing a metric tree like the M-Tree [CPZ97].

11.3.3 kNN-Classification of Multi-Represented objects

Based on the training databases for each representation, we apply the following method of kNN-based classification. To classify a new data object $o = (r_i, \ldots, r_m)$, the kNN sphere $sphere_i(o, k)$ in each representation with $r_i \neq " - "$ is determined. Formally, the $sphere_i(o, k)$ can be described as follows:

$$\begin{aligned} sphere_i(o,k) = \{o_1, \ldots, o_k \mid o_1, \ldots, o_k \in \mathcal{DB}_i \wedge \nexists o' \in \mathcal{DB}_i \setminus \{o_1, \ldots, o_k\} \\ \wedge \nexists \xi, 1 \leqslant \xi \leqslant k : dist_i(o', r_i) \leqslant dist_i(o_\xi, r_i)\} \end{aligned}$$

To combine these kNN spheres and achieve accurate classification, we first of all derive a confidence vector $cv_i(o)$ from each available $sphere_i(o, k)$. Let $c(o)$ denote the correct class of object o and let $d_i^{norm}(u, v)$ be a normalized distance function. Then the confidence vector for an object o with respect to its kNN sphere $sphere_i(o, k)$ for the representation i is defined as follows:

$$cv_i(o) = (cv_{i,1}(o), \ldots, cv_{i,|C|(o)}), \tag{11.1}$$

$$\forall j, 1 \leqslant j \leqslant |C| : cv_{i,j}(o) = \frac{\sum_{u \in sphere_i(o,k) \wedge c(u)=c_j} \frac{1}{d_i^{norm}(o,u)^2}}{\sum_{k=1}^{|C|} cv_{i,k}(o)} \tag{11.2}$$

To normalize our distance function for each representation, we apply the following modification:

$$d_i^{norm}(o,u) = \frac{dist_i(o,u)}{\max_{v \in sphere_i(o,k)} dist_i(o,v)} \tag{11.3}$$

where $dist_i$ is the distance function between two objects in the i-th representation. The normalization in formula 11.3 maps the distance values for each representation to the range $[0,1]$ with respect to the radius of $sphere_i(o,k)$. Thus, the confidence vector of the i-th representation at the j-th position (cf. formula 11.2) is a normalized sum of the inverse quadratic distances.

After we have determined the confidence vectors $cv_i(o)$ for each representation i, we use a weighted linear combination for combining them. Let us note that the combination of confidence vectors to achieve multi-represented classification has been proposed in [Dui02]. However, the used weights in the former approaches do not adjust to the individual classification object. We argue that in order to use each representation in a best possible way, a multi-represented decision rule must weight the influence of all available representations individually for each object.

To achieve this individual weighting, our classification rule is built as follows:

$$Cl_{mr}(o) = \max_{j=1,\ldots,|C|} \sum_{i=1}^{m} w_i \cdot cv_{i,j}(o) \tag{11.4}$$

where m is the number of representations and

$$w_i = \begin{cases} 0 & \text{, if } r_i = \text{"} - \text{"} \\ \frac{1+\sum_{j=1}^{|C|}(cv_{i,j}(o)\cdot \log_{|C|} cv_{i,j}(o))}{\sum_{k=1}^{m}(1+\sum_{j=1}^{|C|}(cv_{k,j}(o)\cdot \log_{|C|} cv_{k,j}(o)))} & \text{, otherwise} \end{cases} \tag{11.5}$$

The idea of our method is that a kNN sphere containing only a small number of classes and several objects of one special class is "purer" than

	Set 1	**Set 2**	**Set 3**	**Set 4**
Name	Enzyme Activity	Metabolism	Transferase	Cell Growth
Number of Goal Classes	267	251	62	37
References to proteins	16815	19639	4086	4401

Table 11.1: Details of the test environments.

a kNN sphere containing one or two objects for each of the classes. Thus, the "purer" a kNN-sphere for a representation is, the better is the quality of the class prediction that can be derived from this representation. To measure this effect, we employ the entropy with respect to all possible classes. The weight is now calculated by normalizing the entropy of its kNN sphere with respect to the entropy of the kNN spheres in all representations. As a result the weights of all representations add up to one. In conclusion, our decision rule for multi-represented objects measures the contribution of each available representation by the entropy in the local kNN spheres of all available representations.

11.4 Experimental Evaluation

Test Bed In order to demonstrate the advantages of our approach, we carried out a versatile experimental evaluation. All algorithms are implemented in Java and were tested on a work station that is equipped with a 1.8 GHz Opteron processor and 8 GB main memory. We used the classification accuracy to measure the effectiveness of algorithms and 5-fold cross-validation to avoid overfitting.

The properties of each test bed are shown in table 11.1. The 4 test

Runtime (in sec.) and Reduction Rate (in %).								
	Set 1 Rep.1	**Set 2 Rep.1**	**Set 3 Rep.1**	**Set 4 Rep.1**	**Set 1 Rep.2**	**Set 2 Rep.2**	**Set 3 Rep.2**	**Set 4 Rep.2**
DBIR	163.0	253.9	8.0	27.5	275.9	1069.6	36.6	119.9
ICF	12,809	15,616	590.0	632.0	93,416	112,248	4,258	3,772
Reduction Rate (in %)								
DBIR	26.1	27.4	33.1	32.0	28.1	22.9	33.8	35.0
ICF	57.0	64.3	71.8	77.7	37.8	46.5	64.0	65.5

Table 11.2: Runtime (in sec.) and reduction rate (in %) reached by DBIR and ICF.

beds consist of 37 to 267 Gene Ontology[Con00] classes. The corresponding objects were taken from the SWISS-PROT [BBA+03] protein database and consist of a text annotation and an amino acid sequence of a protein. In order to obtain a flat class-system with sufficient training objects per class, the original environment was pruned.

We employed the approach described in [DK02] to extract features from the amino acid sequences. The basic idea is to use local (20 amino acids) and global (6 exchange groups) characteristics of a protein sequence. To construct a meaningful feature space, we formed all possible 2-grams for each kind of characteristic, which generated us the 436 dimensions of our sequence feature space. For text descriptions, we employed a TFIDF [Sal89] vector for each description that was built of 100 extracted terms. We used the cosine distance function as distance measure for both representations.

Experimental Results

To demonstrate that DBIR is suitable for large datasets w.r.t. efficiency, we compared the run time needed for data reduction by using DBIR and ICF on single-represented data. As presented in table 11.2, the DBIR outperforms ICF in terms of efficiency, e.g. on the 1st representation of dataset 1, DBIR

Classification Accuracy (in %)								
	Set 1 Rep.1	**Set 2 Rep.1**	**Set 3 Rep.1**	**Set 4 Rep.1**	**Set 1 Rep.2**	**Set 2 Rep.2**	**Set 3 Rep.2**	**Set 4 Rep.2**
kNN	64.43	61.41	72.01	76.2	46.6	43.9	47.48	62.92
kNN DBIR	61.95	60.29	72.56	73.91	44.5	45.5	48.97	56.58
kNN ICF	46.44	35.56	47.92	40.72	37.85	33.21	31.37	34.58

Table 11.3: Classification accuracy (in %) of kNN classifier on: unreduced data, data reduced by DBIR and ICF.

needed only 163 sec. whereas ICF spends 12,809.1 sec. for the data reduction.

To show the effectiveness of DBIR, we compared the classification accuracy achieved by the kNN classifier on unreduced data, data reduced by DBIR and data reduced by ICF (cf. table 11.3). All these experiments were performed on single-represented data. The accuracy achieved by the kNN classifier on data reduced by using DBIR was for all of the datasets comparable to the unreduced dataset. In contrast to these results, the classification accuracy achieved while using ICF was considerably lower. E.g. on the 1st representation of dataset 1, the kNN classification on the data reduced by DBIR reaches 61.95% accuracy, whereas the kNN classification on the data reduced by ICF reaches only 46.44% accuracy. Though the reduction rate achieved by ICF is higher than that of DBIR, the clearly superior accuracy that is achieved by using DBIR indicates that ICF removed important information from the training dataset.

In order to demonstrate the effectiveness of the proposed multi-represented kNN classifier (MR-kNN DBIR), we compared it to the kNN classifier on single representations, naive Bayes (NB) on unreduced single-represented data, NB classification combined by the sum rule and kNN classification combined

Classification accuracy (in %)				
	Set 1	**Set 2**	**Set 3**	**Set 4**
1st Rep., *k*NN DBIR	61.95	60.29	72.56	73.91
2nd Rep., *k*NN DBIR	44.5	45.5	48.97	56.58
1st and 2nd Rep., MR-*k*NN DBIR	**67.65**	**65.17**	**75.52**	**76.8**
1st Rep., NB	43.45	39.95	58.41	41.08
2nd Rep., NB	28.44	22.36	32.87	31.35
1st and 2nd Rep., NB with sum rule fusion	39.64	35.47	51.15	36.03
1st and 2nd Rep., *k*NN classifier fusion by sum rule	62.1	63.18	64.14	74.67
Average classification time per object (in msec.)				
1st Rep., *k*NN DBIR	196.1	198.87	38.22	39.86
2nd Rep., *k*NN DBIR	740.5	907.78	160.42	161.88
1st and 2nd Rep., MR-*k*NN DBIR	1,005.4	1,105.4	198.3	201.6
1st Rep., NB	45.06	43.54	15.4	9.04
2nd Rep., NB	155,91	150,75	48,34	29,62
1st and 2nd Rep., NB with sum rule fusion	206.37	198.3	61.54	36.73
1st and 2nd Rep., *k*NN classifier fusion by sum rule	1,251.3	1,456.2	295.6	316.8

Table 11.4: Classification accuracy (in %) and average classification time per object (in msec.).

by the sum rule. The sum rule described in [Dui02] adds up the confidence vectors delivered by classifiers responsible for single representations. Table 11.4 illustrates the experimental results of this comparison of our approach (MR-kNN DBIR) compared to: *k*NN on single representations reduced by DBIR; Naive Bayes (NB) on single representations and on multiple representations combined by sum rule [Dui02]; *k*NN classifiers combined by sum rule. Our method showed the highest classification accuracy on all datasets and achieved a significant increase of accuracy in comparison to single-represented classification, e.g. on the first set the *k*NN classifier delivered 61.95% accuracy on the first and 44.5% accuracy on the second representation whereas our approach achieved a significantly higher accuracy of 67.65%. NB showed in our experiments low accuracy both on single representations and when combining single NB classifiers employing the sum rule. Our method outperforms also the combination of *k*NN classifiers using the sum rule in all test environments (cf. table 11.4).

11.5 Conclusions

In this chapter, we proposed a novel approach for classifying multi-represented data objects into flat class-systems with huge number of classes that is typical for several advanced applications e.g. in biological or multimedia area. The proposed approach is able to cope with complex objects that might be described by representations that are not necessarily in feature vector form. An important contribution of our method is also a new way of instance reduction to limit the number of employed training objects and thus to speed up classification time without significantly loosing accuracy. Our results indicate that DBIR is capable to reduce the training database faster and provides better accuracy than ICF. To demonstrate the effectiveness of our multi-represented *k*NN classifier, we compared the classification accuracy using related methods and employing classification based on single representations.The results demonstrate that our novel method is capable of outperforming the compared approaches and significantly increases the accuracy by integrating all representations.

Chapter 12

Hierarchical Genre Classification for Large Music Collections using Multiple Representations

In this chapter, we concentrate on hierarchical classification of music content. Section 12.1 gives an introduction to content based classification of music data and addresses the multi-represented and multi-instance characteristics of features extracted from music content. After a short overview of related work in Section 12.2, we introduce a semi-supervised, hierarchical method for reduction of multiple instances and elaborate a framework for hierarchical classification of multi-represented music pieces. Section 12.4 demonstrates results of a versatile experimental evaluation of the proposed techniques. Section 12.5 discusses practical benefits of the developed solution. Section 12.6 concludes this chapter with a short summarization.

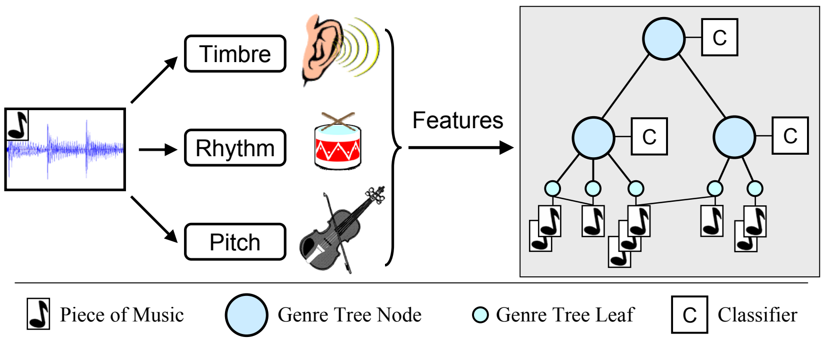


Figure 12.1: Architecture of the proposed framework.

12.1 Introduction

The progress of computer hardware and software technology in recent years made it possible to manage large collections of digital music on an average desktop computer. Often meta information, such as artist, album or title, is available along with the audio file. However, the amount and quality of the available meta information in publicly accessible online databases, e.g. freedb.org, is often limited. This meta data is especially useful when searching for a specific piece of music in a large collection. To organize and structure a collection, additional information such as the genre would be very useful. Unfortunately, the genre information stored in online databases is often incorrect or does not meet the user's expectations.

In this chapter, a content-based hierarchical genre classification framework for digitized audio is presented as sketched in Figure 12.1. It is often problematic to assign a piece of music to exactly one class in a natural way. Genre assignment is a somewhat fuzzy concept and depends on the taste of the user. Therefore, our approach allows multi-assignment of one song to several classes. The classification is based on feature vectors obtained from three acoustic realms namely *timbre*, *rhythm* and *pitch*. Thus, each song is described by multiple representations, each of them containing a set of feature vectors, so called *multiple instances*.

12.2 Related Work

Feature extraction. Timbre features are derived from the frequency domain and were mainly developed for the purpose of speech recognition. The extraction of the timbral texture is performed by computing the short time fourier transform. We use the Mel-frequency cepstral coefficients (MFCCs), spectral flux and spectral rolloff as timbral representations [TC02]. Rhythmic content features are useful for describing the beat frequency and beat strength of a piece of music. In our framework, we use features derived from beat histograms [TC02] as the description of the rhythmic content. Pitch extraction tries to model the human perception by simulating the behavior of the cochlea. Similar to the rhythmic content features, we derive pitch features from pitch histograms which were generated by a multipitch analysis model [TK00].

Genre classification. The general idea of hierarchical classification is that a classifier located on an inner node of the genre tree solves only a small classification problem and therefore achieves more effective results more efficiently than a classifier that works on a large number of flat organized classes. There exist only a few approaches for automatic genre classification of audio data. In [CVK04], music pieces are classified into either rock or classic using k-NN and MLP classifiers. Zhang [Zha03] proposes a method for a hierarchical genre classification which follows a fixed schema and where is only limited support for user-created genre folders. Moreover, the above mentioned hierarchical classification methods do not take full advantage of multi-instance and multi-represented music objects. In contrast, our approach handles such rich object representations as well as an arbitrary genre hierarchy, and supports multi-assignment of songs to classes.

Hierarchical Classification. The use of class hierarchies to improve large scale classification problems has predominantly been applied in text classification. Several approaches have been introduced picking up this idea. The

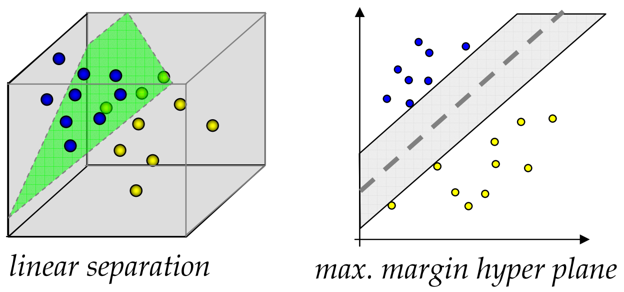


Figure 12.2: Basic idea of Support Vector Machine (SVM).

authors of [KKPS04b] investigated multiple representations of objects in the context of hierarchical classification and proposed a so called *object adjusted weighting* for linear combination of MR objects.

Support Vector Machines. In recent years, *support vector machines* (SVMs) [CV95] have received much attention offering superior performance in various applications. For example, [WLCS04] presents a fusion technique for multimodal objects. Basic SVMs use the idea of linear separation of two classes in feature space and distinguish between two classes by calculating the maximum margin hyperplane between the training examples of both given classes as illustrated in Figure 12.2. To employ SVMs for distinguishing more than two classes, several approaches were introduced [PCST99]. In order to handle sets of feature vectors in SVMs so called kernel functions were introduced [GFKS02a]. A weakness of multi-instance kernels is the need to calculate distances between all instances, i.e. $O(n^2)$ single distance calculations are required in order to compare two multi-instance objects with n instances. Thus, multi-instance kernels seem to be unsuitable for solving large scale classification problems in music collections.

Classifier Fusion. All methods of clasifier fusion or ensemble learning discussed in Section 2.2.2 assume that a classifier provides reliable values of the posteriori probabilities for all classes. In contrast to these solutions, we propose a method that calculates a object adjusted weighting that reflects the correctness of each particular class decision.

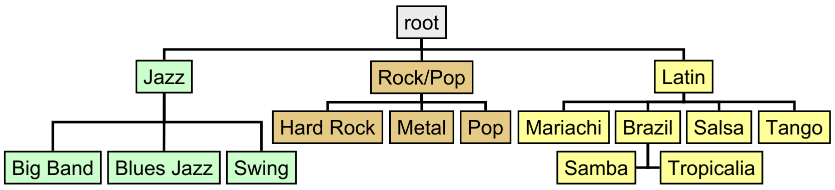


Figure 12.3: An example genre hierarchy.

Instance Reduction Techniques. As mentioned above, a piece of music is usually described by a set of feature vectors and is an multi-instance object. The number of instances can vary from tens to hundreds per second, i.e. a song is represented by 10,000 to 50,000 feature vectors. In order to handle such multi-instance objects two classes of instance reduction techniques can be distinguished, namely higher-order and first-order (cf. Section 2.1.5 in Chapter 2 for details). Both first and higher-order techniques reduce the multi-instance object to a small set of feature vectors. Thus, using the reduced representations of the multi-instance object requires the application of kernel functions for SVMs. In context of large databases, the use of kernel functions seems impracticable for efficient classification.

12.3 Efficient Hierarchical Genre Classification

In this section, we describe our approach for classifying large collections of music pieces in a genre taxonomy (cf. Figure 12.3). Since a music piece is described by a set of feature vectors, we first describe a novel hierarchical semi-supervised technique for instance reduction. The reduced descriptions are used afterwards for hierarchical classification of music pieces with SVMs. Furthermore, we use object adjusted weighting in order to take advantage from multiple representations.

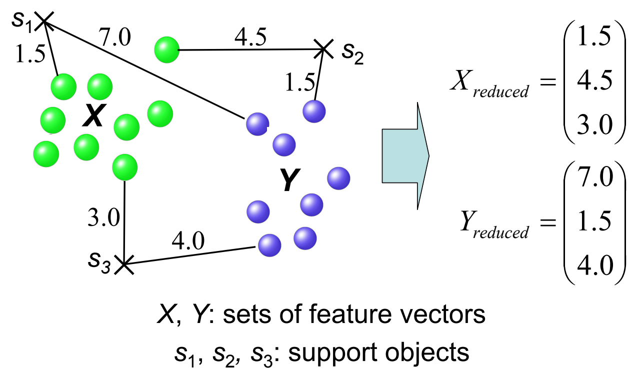

s1
s2
s3
1.5
7.0
4.5
1.5
3.0
4.0
X
Y
X_reduced = (1.5, 4.5, 3.0)
Y_reduced = (7.0, 1.5, 4.0)
X, Y: sets of feature vectors
s1, s2, s3: support objects

Figure 12.4: Instance reduction with help of support objects.

12.3.1 Hierarchical Instance Reduction.

Let $\mathcal{DB}$ be a set of music objects. We argue that an MI object $X = \{x_1, \ldots, x_n\} \in \mathcal{DB}$ can be described by a vector $X_{reduced}$ containing minimal distances to a given set of so called *support objects* $S = \{s_1, \ldots, s_m\}$ where $m \ll n$. Formally,

$$X_{reduced} = (\min_{x_i \in X} dist(x_i, s_1), \ldots, \min_{x_i \in X} dist(x_i, s_m)).$$

The set S can either be calculated by a random selection of m instances from DB, or it is possible to choose each $s_i \in S$ as a centroid of a clustering that can be calculated on a small sample of instances from DB. An example for the instance reduction is illustrated in Figure 12.4.

The number of elements in $X_{reduced}$ may still be too large for solving the classification problem efficiently. Thus, we propose to exploit the hierarchical organization of classes and to select only a small subset $S_N \subseteq S$ for each inner node N of the genre taxonomy. The elements of S_N should be selected so that the subclasses C_N of N can be distinguished in the best possible way. Therefore, the subset of support objects is individual for each inner node N.

To calculate S_N we suggest to apply a semi-supervised method based on the *information gain criterion*. Let $T(C_N)$ be a set of all training objects belonging to C_N. The domains $D(s_i)$ are discretized by using the method

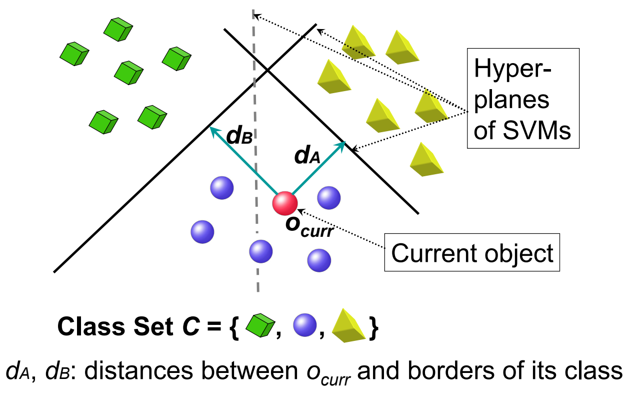

Hyper-planes of SVMs
d_B
d_A
o_{curr}
Current object
Class Set C = { 🟩, ●, ▲ }
d_A, d_B: distances between o_{curr} and borders of its class

Figure 12.5: Border distance based derivation of weights for a multi-represented object.

described in [FI92]. After discretization the information gain criterion for each attribute can by calculated by

$$InfoGain(s_i, T(C_N)) = H(T(C_N)) - \sum_{t \in T(C_N)} \frac{|t|}{|T(C_N)|} \cdot H(t),$$

where $H(t)$ denotes the entropy. Finally, S_N is calculated as follows: $S_N = \{s_j \in S \,|\, |S_N| = k \wedge \forall s_j \in S_N \forall a \in S : InfoGain(a, T(C_N)) \leq InfoGain(s_j, T(C_N))\}$. After that, S_N is used for training and classification on the node N.

12.3.2 Hierarchical Genre Classification by Using Multiple Representations.

A *two layer classification process* (2LCP) handles the hierarchical classification problem on each inner node N of the genre taxonomy. This process acts as a guidepost for the hierarchical classification. We train SVMs in the first layer of the 2LCP that distinguishes only single classes C_{single} in each representation. Since standard SVMs are able to make only binary decisions we apply the so-called one-versus-one (OvO) approach (cf. Figure 12.5) in order to make a classification decision for more than two classes. We argue that for our application the OvO approach is best suitable because the voting vectors

Φ_i provided by this method are a meaningful intermediate description that is useful for solving the multi-assignment problem in the second layer of our 2LCP. In order to perform the multi-assignment we take advantage of the class properties in our application domain. We limit the possible class combinations to a subset $C_{combi} \subset 2^{C_{single}}$ because there exist several combinations that do not make sense, e.g. a piece of music belonging to the class 'salsa' is very implausible to be also in the class 'metal'. For this purpose, we only take those $c \in 2^{C_{single}}$ into account, which occur in the training set.

The SVM classifier in the second layer of the 2LPC uses an aggregation of the voting vectors Φ_i from the first layer of the 2LPC as input to assign an object to a class $c \in C_N = C_{single} \cup C_{combi}$. The second task that is handled by the classifier in the second layer is the aggregation of multiple representations. The voting vectors $\Phi_1, \ldots, \Phi_k$ provided by the first layer SVMs for each representation $R_1, \ldots, R_k \in R$ are aggregated by using a weighted linear combination $V = \sum_{i=1}^{k} \omega_i \Phi_i$. Then V is used as the input for the classifier in the second layer. The weights ω_i in the combination are calculated by using object adjusted weighting. The intuition behind the object adjusted weighting is that the current object o_{curr} used in training or to be classified needs to have a sufficient distance from any of the other classes. Furthermore, the closer surrounding of the hyperplane is treated in a more sensitive way. More formally, let c_j be the class of o_{curr} determined by majority vote in Φ_i, then $\omega_i = sigmoid(\min_{c_i \in C_{single} \wedge c_i \neq c_j} dist(o_{curr}, HyperPlane(c_j, c_i)))$, where $HyperPlane(c_j, c_i)$ denotes the maximum margin hyperplane separating the classes c_j and c_i and $sigmoid$ denotes sigmoid function defined as $sigmoid(x) = \frac{1}{1+exp(\alpha \times x+\beta)}$. A suitable optimization algorithm (e.g. Levenberg-Marquardt algorithm [Lev44]) is used to determine the parameters α and β that minimize the least squares error for the sigmoid target function $accuracy(o) = \frac{1}{1+exp(\alpha \times x+\beta_j)}$ given the observed pairs of confidence ranges and classification accuracy. Figure 12.5 depicts an example of weight calculation where the weight ω should be set according to the value of d_A.

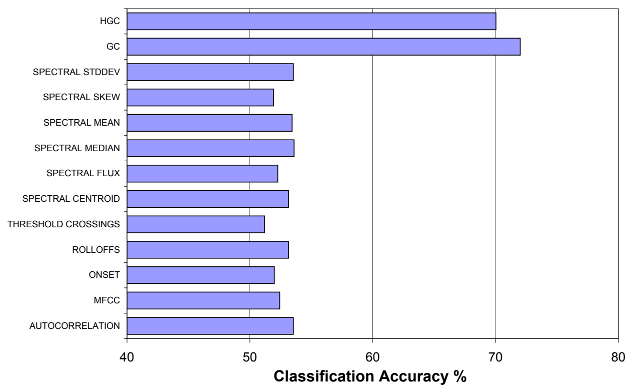


Figure 12.6: Accuracy for classification on single- and multi-representations.

12.4 Experimental Evaluation

We implemented our approach in Java 1.5 and performed all experiments on a Pentium IV workstation equipped with 2 GByte main memory. The genre hierarchy depicted in Figure 12.3 was used in all following experiments. A music collection consisting of almost 500 songs with an average duration of 4 minutes 14 seconds was the basis for the classification experiments, which results in approximately 30 songs per class. Depending on the representation, we extracted 30 to 200 features per second. We performed 10-fold cross-validation for evaluating the classification accuracy. In the following, we present the results of our experiments with particular emphasis to efficiency and effectiveness.

Effectiveness. In the first experiment, we compared the quality of GC on multiple, and HGC on single and multiple representations. Figure 12.6 depicts the experimental results. When working with multiple representations, our HGC approach (70.03%) achieves higher classification accuracy than using a single representation only. Furthermore, the classification accuracy of HGC is comparable to that of the flat GC approach (72.01%).

In the next experiment, we investigated how the classification accuracy

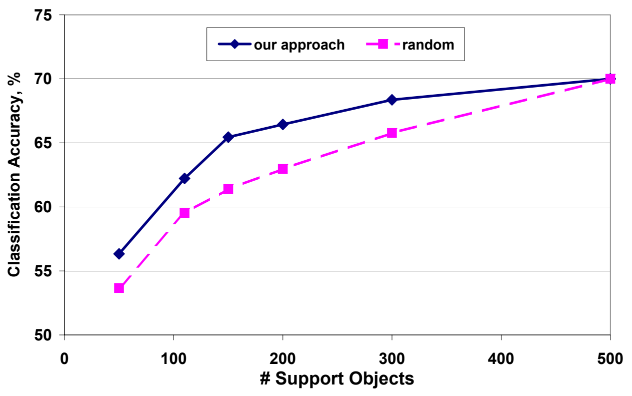


Figure 12.7: Accuracy for classification on single- and multi-representations.

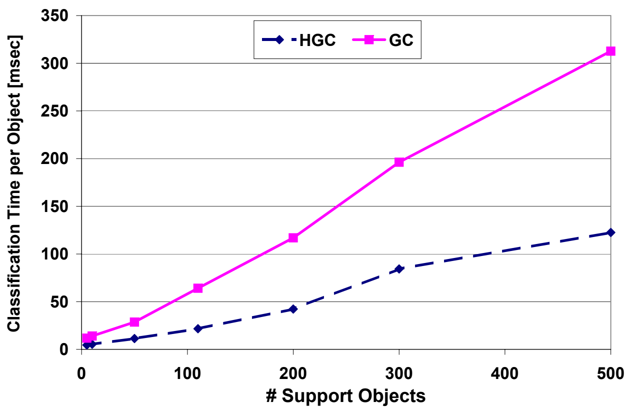


Figure 12.8: Classification time per object.

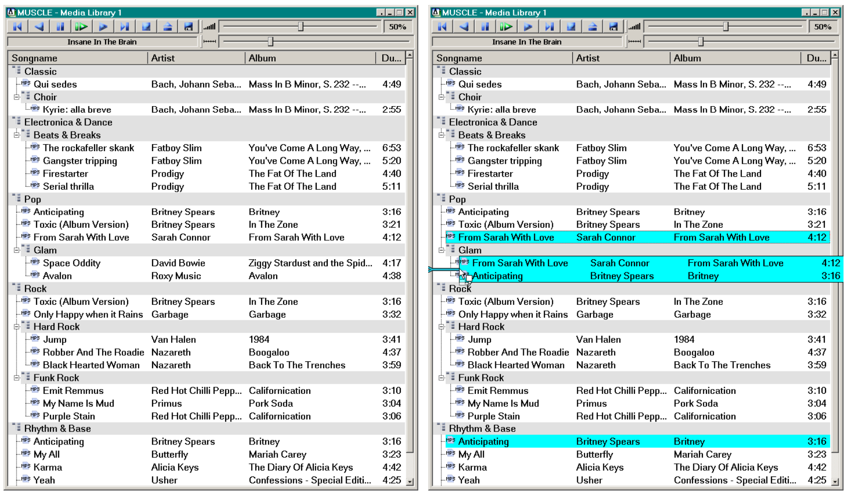
of our approach is influenced by the number and the choice of the support objects. For choosing S_N, we either randomly picked the support objects or applied our strategy described in Section 12.3. The experimental results are depicted in Figure 12.7 and show that our approach always outperforms the random selection. For both approaches, the accuracy increases with an increasing number of support objects. However, especially for a low number of support objects, the random approach achieves a lower accuracy compared to our method. For a high number of support objects, both approaches yield a similar classification accuracy.

Efficiency. In a last experiment, we examined the runtime performance of GC and HCG for a varying number of support objects. As depicted in Figure 12.8, the runtime increases with an increasing number of support objects. The higher the number of support objects, the larger the runtime difference. Altogether, our approach achieves a good trade-off between the quality of the result and the required runtime when using 300 support objects.

12.5 Practical Benefits

A framework, called “MUSCLE:Music Classification Engine with User Feedback” (cf. [BKK+06]), with settings similar to described in this chapter is implemented in C/C++ and runs on the Windows platform. Its hierarchical playlist acts as a jukebox. The installation archive of MUSCLE contains a default genre taxonomy including the necessary training data in the form of feature vectors for each song. This data is used in the demonstration. Using aggregated information such as feature vectors makes it possible to share the training data without having to distribute the underlying music data. Classes and training data in the genre taxonomy can be deleted, moved or added by the user. When the user commits the changes of the class hierarchy or of the corresponding training data, MUSCLE trains the affected classifiers. Note that usually only a small subset of the entire classifier hierarchy has to be trained because a modification at a node requires a partial adaptation of the node and all parent nodes only. It is also possible to start the training automatically after each modification or to run the training in the background. When the user is satisfied with the training setup, a folder to automatically classify all contained songs can be selected.

Fig. 12.9 illustrates MUSCLE’s user interface. In the main window the playlist containing the classification result in form of a genre tree is displayed. An example for a multiple assignment of the song ’Anticipating’ to the classes



MUSCLE - Media Library 1

Insane In The Brain

50%

Songname	Artist	Album	Du...
Classic			
Qui sedes	Bach, Johann Seba...	Mass In B Minor, S. 232 --...	4:49
Choir			
Kyrie: alla breve	Bach, Johann Seba...	Mass In B Minor, S. 232 --...	2:55
Electronica & Dance			
Beats & Breaks			
The rockafeller skank	Fatboy Slim	You've Come A Long Way, ...	6:53
Gangster tripping	Fatboy Slim	You've Come A Long Way, ...	5:20
Firestarter	Prodigy	The Fat Of The Land	4:40
Serial thrilla	Prodigy	The Fat Of The Land	5:11
Pop			
Anticipating	Britney Spears	Britney	3:16
Toxic (Album Version)	Britney Spears	In The Zone	3:21
From Sarah With Love	Sarah Connor	From Sarah With Love	4:12
Glam			
Space Oddity	David Bowie	Ziggy Stardust and the Spid...	4:17
Avalon	Roxy Music	Avalon	4:38
Rock			
Toxic (Album Version)	Britney Spears	In The Zone	3:16
Only Happy when it Rains	Garbage	Garbage	3:32
Hard Rock			
Jump	Van Halen	1984	3:41
Robber And The Roadie	Nazareth	Boogaloo	4:37
Black Hearted Woman	Nazareth	Back To The Trenches	3:59
Funk Rock			
Emit Remmus	Red Hot Chilli Pepp...	Californication	3:10
My Name Is Mud	Primus	Pork Soda	3:04
Purple Stain	Red Hot Chilli Pepp...	Californication	3:06
Rhythm & Base			
Anticipating	Britney Spears	Britney	3:16
My All	Butterfly	Mariah Carey	3:23
Karma	Alicia Keys	The Diary Of Alicia Keys	4:42
Yeah	Usher	Confessions - Special Editi...	4:25

MUSCLE - Media Library 1

Insane In The Brain

50%

Songname	Artist	Album	Du...
Classic			
Qui sedes	Bach, Johann Seba...	Mass In B Minor, S. 232 --...	4:49
Choir			
Kyrie: alla breve	Bach, Johann Seba...	Mass In B Minor, S. 232 --...	2:55
Electronica & Dance			
Beats & Breaks			
The rockafeller skank	Fatboy Slim	You've Come A Long Way, ...	6:53
Gangster tripping	Fatboy Slim	You've Come A Long Way, ...	5:20
Firestarter	Prodigy	The Fat Of The Land	4:40
Serial thrilla	Prodigy	The Fat Of The Land	5:11
Pop			
Anticipating	Britney Spears	Britney	3:16
Toxic (Album Version)	Britney Spears	In The Zone	3:21
From Sarah With Love	Sarah Connor	From Sarah With Love	4:12
Glam			
From Sarah With Love	Sarah Connor	From Sarah With Love	4:12
Anticipating	Britney Spears	Britney	3:16
Rock			
Toxic (Album Version)	Britney Spears	In The Zone	3:16
Only Happy when it Rains	Garbage	Garbage	3:32
Hard Rock			
Jump	Van Halen	1984	3:41
Robber And The Roadie	Nazareth	Boogaloo	4:37
Black Hearted Woman	Nazareth	Back To The Trenches	3:59
Funk Rock			
Emit Remmus	Red Hot Chilli Pepp...	Californication	3:10
My Name Is Mud	Primus	Pork Soda	3:04
Purple Stain	Red Hot Chilli Pepp...	Californication	3:06
Rhythm & Base			
Anticipating	Britney Spears	Britney	3:16
My All	Butterfly	Mariah Carey	3:23
Karma	Alicia Keys	The Diary Of Alicia Keys	4:42
Yeah	Usher	Confessions - Special Editi...	4:25

(a) Multi-Assignment of Songs **(b)** User Feedback

Figure 12.9: MUSCLE User Interface

'pop' and 'rhythm & base' can be seen in Fig. 12.9(a). In case the user wants to manually adjust the genre assignment of a song, entries can be re-arranged using drag & drop as shown in Fig. 12.9(b).

12.6 Conclusions

In this chapter, we introduced a novel semi-supervised, hierarchical *instance reduction* technique which enables us to use only a small number of relevant features for each classifier. Furthermore, we elaborated a framework for hierarchical music classification using multiple representations consisting of multiple instances. We showed that our hierarchical classification can compete with a flat class system in terms of effectiveness and greatly surpasses it in terms of efficiency. An implementation of our framework has been demonstrated recently [BKK$^+$06].

Part IV

Conclusions

Chapter 13

Summary of Contributions

In recent years, advanced database systems have emerged. They are necessary because of the demand for storage, management and retrieval of large amounts of data in application areas such as molecular biology, biometrics, multimedia, and location based services. In contrast to conventional database systems, users of advanced database systems focus on similarity search and data mining tasks. Based on an extensive analysis of objects and properties typical for advanced database systems, we have determined three requirements — uncertain, multi-instance, and multi-represented data — which so far were insufficiently considered in similarity search and data mining areas. This thesis presents novel similarity search, clustering, and classification approaches that are designed to handle uncertain, multi-represented and multi-instance descriptions of data to achieve enhanced results. This chapter summarizes the major theoretical and practical contributions of this work.

13.1 Preliminaries (Part I)

In Section 1.1 of Chapter 1 we considered advanced database systems and analyzed characteristics of data objects and tasks that are typical for advanced database systems. In Sections 1.2 and 1.3 we introduced state-of-the-art similarity search and data mining techniques. Based on the typical characteristics of data objects and tasks in advanced database systems, new challenges for similarity search and data mining techniques were elaborated in Section 1.4. After a short outline of this thesis in Section 1.5, in Chapter 2 we surveyed important previous work in similarity search, clustering and classification areas related to techniques proposed in this thesis.

13.2 Similarity Search Techniques (Part II)

In the first chapter of Part II we introduced the *Gaussian uncertainty model for identification queries* on inexact, probabilistic feature vectors. This model extends feature vectors by an additional uncertainty value for each dimension, associating each feature vector with a multivariate Gaussian distribution. Applications of uncertain data are biometric identification using fingerprints, facial images, speech patterns, etc. Furthermore, this model is applicable for object identification in multimedia databases.

Based on the probabilistic model, we proposed two novel query types – *Threshold Identification Queries (TIQ)* and *k-Most Likely Identification Queries (k-MLIQ)*. To speed up query types such as TIQ or k-MLIQ, we developed the *Gauss-tree*, a balanced index structure from the R-tree family which does not index the Gaussian curves as spatial objects but the parameter space of the means and variances of the Gaussians. In order to adapt the defined index structure for handling Gaussians, *the algorithms for both insertion and split* were proposed. The split and the insertion algorithm of the Gauss-tree is based on the density hull curve of a node. If the integral of this curve is rather small, the indexed Gaussians are rather similar. Thus, the split algorithm favors splits resulting in nodes which have a rather small integral over the density hull. In our experimental evaluation, we observed the superior quality of the query result when using probabilistic feature vectors as well as the efficiency when using the Gauss-tree.

In Chapter 4, we proposed an efficient technique for high performance video retrieval. This approach is based on a summarization technique by using probabilistic feature vectors, i.e. Gaussian probability density functions. For the storage and efficient retrieval of probabilistic feature vectors, the specialized index structure, Gauss-tree, see Chapter 3, was extended to handle *sets of probabilistic feature vectors*. Every video clip in the database is associated with a set of probabilistic feature vectors. Query video clips are also transformed either into a set of conventional feature vectors or into a set of probabilistic feature vectors. In both cases, query processing involves *matching of sets of vectors to sets of vectors*. We defined two kinds of set-valued queries, *set-valued probabilistic ranking queries* and *set-valued probabilistic threshold queries*, and proposed *efficient algorithms for query processing* on top of the Gauss-tree. An extensive experimental evaluation using 900 video clips demonstrated the superiority of our approach with respect to both accuracy as well as efficiency of retrieval.

In Chapter 5, we suggested *the Gaussian uncertainty model* for describing *uncertain spatial objects*. This model describes an uncertain data object as a probabilistic feature vector consisting of a mean value and a standard deviation for any uncertain feature value. Assuming a Gaussian density distribution based on these parameters, we can now determine the probability of any data object being contained in a certain interval or (hyper-) rectangle. Applications for uncertain spatial objects are databases of moving objects and sensor networks where the exact feature value cannot be constantly monitored. To query databases of uncertain spatial objects, we can pose probabilistic queries like probabilistic threshold queries (PTQs). A PTQ retrieves all data objects in a database that are contained in the query rectangle with a larger probability than some probability threshold t. Since it is often difficult to determine the threshold value, we introduced *probabilistic* *ranking queries (PRQs)* which retrieve the k data objects in a database that are contained in the query rectangle with the highest probability. To answer both types of queries within a reasonable time, we applied the Gauss-tree, see Chapter 3. The idea of the Gauss-tree is to index the parameter space of the probabilistic feature vectors (pfv) in the database. A node in the Gauss-tree contains pfvs having mean values and standard deviation contained in a certain mean range and a certain range of standard deviations. Based on these ranges, a *conservative approximation* for a node and a given query rectangle can be calculated. This tight approximation is the basis of the described *algorithms for answering PTQs and PRQs.* In our experimental evaluation, we compared the Gauss-tree on both types of queries with three comparison partners on one artificial and two real-world datasets with artificial uncertainty. The results demonstrated that the Gauss-tree achieves a query performance which is comparable to state-of-the-art methods on PTQs. For the new query type of PRQs, the Gauss-tree clearly outperforms established methods which were modified to answer PRQs as well.

Similarity search in multimedia databases can be improved by using multiple representations of the multimedia objects. When searching for similar videos, e.g. we can use audio features such as rhythm and pitch as well as video features such as color histograms and textures. In Chapter 6, we presented a method for *effective similarity search* in multimedia databases that takes *multiple representations* of the database objects into account. This technique exploits the fact that it is often beneficial to *summarize multiple instances* of a multimedia object, e.g. video, in order to achieve a higher efficiency during query processing. We introduced *four quality scores* to derive the weight for each representation. Our concepts are independent of the underlying summarization method and compute a weight for each summarization vector of each representation for each object separately. Using these weighting factors, we further showed how well-known distance measures for multi-instance multimedia objects can be combined w.r.t. multiple representations. In our experiments, we evaluated the proposed methods and demonstrated the benefits of our approach.

13.3 Data Mining Techniques (Part III)

In Chapter 7, we proposed a *distributed clustering algorithm* that achieves *privacy perceivation* by *using an uncertain description* of the data. Our method applies the EM algorithm at the local sites generating a model containing a *set of Gaussian distributions*. We also proposed a *merge step* of the local Gaussians. Compared to recent approaches for pure distributed clustering, our method enables us to set an arbitrary level of privacy and dramatically reduces the transfer cost. Our experimental evaluation demonstrated robustness, efficiency, and applicability of the proposed technique.

In Chapter 8, we described an approach for the *statistical clustering of multi-instance objects*. Our approach models *instances as members of concepts* in some underlying feature space. Each concept is modeled by a statistical process in this feature space. A multi-instance object can now be considered as the result of selecting several times a concept and generating an instance with the corresponding process. Clusters of multi-instance objects can now be described as *multinomial distributions over the concepts*. In other words, different clusters are described by having different probabilities for the underlying concepts. An additional aspect is the length of the multi-instance object. To derive multi-instance clusters corresponding to this model, we introduced a three-step approach. The first step derives a mixture model that describes concepts in the instance space. The second step finds a good initialization for the target distribution by subsuming each multi-instance object by a so-called confidence summary vector and afterwards clustering these confidence summary vectors by using the k-means method. The final step employs an EM clustering step, optimizing the distribution for each cluster of multi-instance objects. To evaluate our method, we compared our clustering approach to clustering multi-instance objects with the k-medoid clustering algorithm PAM for three different similarity measures that are appropriate for multi-instance objects. The results demonstrate that the found clustering model offers better cluster qualities w.r.t. to the provided reference clusterings.

In Chapter 9, we proposed COSMIC (**Co**nceptually **S**pecified **M**ulti-**I**nstance **C**lusters), a method for *deriving concept lattices* from *multi-instance datasets*. COSMIC describes concepts of multi-instance objects by sets of so-called *cluster attributes* (CAs). A CA is a common pattern in the data space of instances that might be used to characterize at least *MinPts* multi-instance objects. To use CAs on different abstraction levels and thus to be less dependent on the parameter setting, COSMIC employs a hierarchy of CAs. The *CA hierarchy* is calculated by employing *density-based hierarchical clustering* while considering that a CA has to describe instances from at least *MinPts* multi-instance objects. The second step of COSMIC extracts a concept lattice along with the CA hierarchy used for the concept descriptions. In our experimental evaluation, we compared COSMIC to two distance-based approaches for clustering multi-instance data on five real-world datasets. The results demonstrated that COSMIC generates more precise clusterings w.r.t. a reference class set. Additionally, we showed that COSMIC scales well to even larger datasets and is very insensitive to the choice of its two main parameters *MinPts* and *MinSup*.

In Chapter 10, we discussed the problem of the *density-based clustering of multi-represented objects* and suggested two novel clustering techniques. The idea of our first approach is to combine the information of all different representations as early as possible and as late as necessary. To do so, we *adapted the core object property* proposed for DBSCAN. To decide whether an object is a core object, we use the local ε-neighborhoods of each representation and combine the results to a *global neighborhood*. Based on this idea, we proposed two different methods for varying applications. For sparse data, we introduced *the union-method* that assumes that an object is a core object if k objects are found within the union of its local ε-neighborhoods. Respectively, we defined *the intersection-method* for data where each local representation yields rather big and unspecific clusters. Therefore, the intersection-method requires that at least k objects are within the intersection of all local ε-neighborhoods of a core object.

Furthermore, we discussed the problem of *hierarchical density-based clustering of multi-represented objects* having *arbitrary semantics*. Since each representation might have a different meaning, we first divided the representation spaces into two basic types, the *precision* and the *recall spaces*. After drawing elementary conclusions about the basic types and how they should be *combined by using union and intersection operators*, we introduced *combination trees* for describing arbitrary combinations of multiple representations. To cluster multi-represented objects w.r.t. a combination tree, we adapted the hierarchical clustering algorithm OPTICS to the multi-represented setting. In our experimental evaluation, we introduced an *entropy based quality measure* that compares a given clustering with noise to a reference clustering. Employing this quality measure, we demonstrated that the union method was most suitable to overcome the sparsity of a given protein dataset. To demonstrate the ability of the intersection method to increase the cluster quality, we applied it to a set of images using two different similarity models. Experimental results with our hierarchical multi-represented clustering approach showed the improvement of clustering results for an image dataset that is described by four representations as well as for protein datasets.

In Chapter 11, we proposed a novel approach for *classifying multi-represented objects* into flat class-systems with *many classes*. The proposed method addresses the three following requirements that are frequent in the advanced database systems: (1) multi-represented objects, (2) representations that are *not necessarily in feature vector form*, (3) *large class sets*. To cope with these requirements, our new method for classification of multi-represented objects employs kNN classification because this approach is naturally able to handle the last two requirements. The contribution of our method is a new way of *training instance reduction based on a density-based paradigm* to limit the number of employed training objects and thus speed up classification time without significantly loosing accuracy. To integrate the information of several representations, we present a new decision rule that employs *a weighted combination of confidence values* to derive a class prediction. The idea of the used weighting is *to measure the entropy* of each kNN sphere. Thus representations are weighed in a different way for different data objects. In our experimental evaluation, we compared our new density-based instance reduction technique to one of the best performing instance reduction techniques so far. Our results indicated that the proposed method is capable of reducing the training database faster and provides better accuracy when compared to competing methods. To demonstrate the effectiveness of our multi-represented kNN classifier, we compared the classification accuracy using related methods and employed classification based on single representations. The results demonstrate that our new method can outperform the compared approaches and significantly increases the accuracy by integrating all representations.

In Chapter 12, we introduced a novel method for the *reduction of large multi-instance objects* and *a framework for hierarchical classification* of multimedia objects, e.g. music pieces. Our method uses *multiple representations* consisting of *multiple instances.* We showed that our hierarchical classification can compete with a flat class system in terms of effectiveness and greatly surpasses it in terms of efficiency.

Chapter 14

Future Work

At the end of this thesis let us consider possible further directions for research which have been motivated by novel techniques for similarity search, clustering and classification developed in this thesis. First, we discuss promising enhancements of the methods proposed in this work. In addition, we sketch our vision of the future of similarity search and data mining techniques for advanced database systems.

Short-term Considerations: Enhancements of the Proposed Methods. In Chapters 3 through 5 we proposed and employed an index structure for probabilistic feature vectors based on a hierarchical paradigm. An interesting direction for future work is to investigate the storage of probabilistic feature vectors using paradigms different from hierarchical index structures such as vector approximation. Uncertain spatial, biometrical and multimedia data are often generated and stored at different geographical locations. Therefore, a very interesting direction for future research could be the development of approaches indexing distributed uncertain data in general, and to parallelize the Gauss-tree for a distributed database environment in particular.

In Chapter 6, we developed a novel method for an effective similarity search in multimedia databases using multiple representations. Current research aims also at employing as much information from multiple representations as possible in order to achieve better precision and recall (cf. methods described in [BKS+04, SJL+03a, CLC98]). However, combining multiple feature representations yields various problems. Not all feature transformations are suitable for each dataset. Additionally, even if a representation is not very well suited for a given dataset in general, it is still possible that the distances between some of the objects still model similarity rather well. Therefore, a system using multiple representations should consider the quality of each object in each representation. Thus, an interesting direction for further research is developing methods based on similarity and dissimilarity estimates. While the similarity estimates represent the likelihood that the compared objects are similar, the dissimilarity estimates indicate that the compared objects are dissimilar. Based on both types, we can estimate the likelihood over all representations that the compared objects are indeed similar or not. Furthermore, it seems to be promising that we derive both types of estimates for each representation without manually labeling pairs of similar objects.

In Chapters 7 and 8 we proposed to employ Gaussian distributions in order to describe the uncertainty of objects or to model underlying data. One of the possible promising directions for future work is to examine the use of other distribution functions instead of the normal distribution. For instance, we could develop statistical processes that describe more complex objects than multi-instance objects and employ these processes for EM clustering as well.

In Chapter 10, we discussed precision and recall spaces. For future work, it is interesting to find the best possible way to quantify the usability of representations as precision or recall spaces. A very interesting research direction is the development of a general theory for describing optimal combination trees.

Another interesting idea would be to investigate the use of various index structures to speed up classification methods like a method described in Chapter 11 or in Chapter 12. Furthermore, it would be an enhancement to apply our methods introduced in Chapters 11 and 12 to biometric identification. Biometric identification yields several individual challenges like the combination of different classification methods. For example, facial features can be checked by kNN classification. However, in order to recognize a person by speech pattern, other ways like hidden Markov models are reported to provide better accuracy. Thus, a flexible model should allow different classification algorithms. Another interesting direction is to speed up classification by employing only some of the representation. For example, it might be unnecessary to query the sequence database if the text database provides sufficient confidence.

We suggested in this work two classification approaches which deal with complex objects described by multiple representations, modeling various aspects, and using various feature transformations (cf. Chapter 11 and Chapter 12). To integrate all the information from different representations into the classification, we trained a classifier on each representation and combined the results based on the local class probabilities. It is an interesting idea to derive confidence estimates for each of the classifiers, reflecting the correctness of the local class prediction and use the prediction having the maximum confidence value. The confidence estimates can be based on the distance to the class border as proposed in Chapter 12. Then we can derive the confidence estimates for various types of classifiers like support vector machines, k-nearest neighbor classifiers, Bayes classifiers and decision trees. This approach promises the following two advantages. First, we can employ different classifier types for different representations. Second, the comparability of classification results is guaranteed because we apply the same approach on each classifier in order to estimate the confidence.

Long-term Considerations: Coping with Increasing Complexity of

Real-World Objects. Modern automated methods for measurement, collection, and analysis of data in all fields of science, and industry are providing more and more data with drastically increasing structure complexity. This growing complexity is justified on the one hand by the need for a richer and more precise description of real-world objects, and on the other hand by the rapid progress in measurement and analysis techniques allowing versatile exploration of objects. In order to manage the huge volume of such complex data, advanced database systems are employed. Thus, advanced database systems provide and manage manifold information concerning all kinds of real-world objects, ranging from customers and molecules to shares and patients.

Traditionally, relational databases keep this information in the form of attributes from a certain range of possible domains, usually as numbers, dates, strings, or restricted to a certain list of values. Object-relational databases even allow the user to define types that model arbitrary objects. In view of the fact that the manual analysis of enormous volumes of complex data collected in a database is practically infeasible, there is an ever growing need for similarity search and data mining techniques that are able to discover novel, interesting knowledge in this complex and voluminous data. Quite some efforts lead in various directions of coping with complexity of data objects like similarity search and data mining techniques for uncertain, multi-instance and multi-represented objects that were developed and discussed this thesis.

While modeling the world obviously creates a merely simplified representation, considering the complexity of the objects as adequate as possible remains a worthwhile goal for all directions of science. In computer science, the concept of "object-oriented modeling" describes complex objects in a simple and thoroughly formalized manner. Here, attributes of an object may be primitive types or objects themselves. Object-oriented and also object-relational databases are able to present collections of such objects. It seems highly desirable to be able to directly mine on these objects instead of mining only parts of them (like their numerical attributes or numerical models of their complex attributes). In recent years, many steps were taken to mine objects modeled as multi-represented, multi-relational or multi-instance data. In some respects, these approaches are generalizations of former approaches on unstructured data. On the other hand, the very same approaches could be understood as adjustments to certain more general, but not universal types of representations. We envision similarity search and data mining being universalized to tackle with truly general objects. However, all these methods consider static properties of objects. The picture of "object-oriented modeling" does also include a modeling of behavior of objects, called "methods", i.e. dynamical properties. Furthermore, sequence diagrams or activity diagrams model the chronology of behavior patterns. Indeed, the behavior of software is a common data mining task (cf. e.g. [LYH06, LYY+05]). Some steps towards directly mining object-oriented systems can be found e.g. in [KDTM06].

Recently, domain experts seek ways to extract the important features of an object. Thus, representing complex objects by means of simple objects like numerical feature vectors could be understood as a way to incorporate domain knowledge into the similarity search and data mining process. In the progress to generalized similarity search and data mining, one should not disregard the advances made so far. Incorporating domain knowledge as naturally as possible facilitates meaningful results of similarity search and data mining. However, the specific way to make use of the domain knowledge of experts should also be generalized to keep pace with more complex ways of mining complex objects.

Furthermore, the knowledge specific to a certain domain is increasing in amount and complexity. Usually it cannot be surveyed by a single human expert anymore. Therefore, the communities provide their knowledge often in databases or knowledge bases. Thus, in the future, similarity search and data mining algorithms should be able to take reliable domain knowledge, which is available in databases, automatically into account in order to improve their effectiveness.

In order to process complex objects, distributed similarity search and data mining seems to gain in importance [LKBR06]. Several application domains consider the same complex object according to the same characteristics at different locations and/or at different times (e.g. a patient can consult different doctors, or a continuous observation of a star is only possible by involving several telescopes around the world). On the other hand, similarity search and data mining algorithms requires significantly more computation power on complex objects than on data given by feature vectors. Finally, not all participants in a joint activity would like to share all of their collected data, possibly in order to protect the privacy of their customers. Thus, there is a growing need for distributed, privacy preservation exploration and analysis algorithms for complex data like the method proposed in Chapter 7.

List of Figures

List of Tables

References

[ABKS99] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. OPTICS: Ordering points to identify the clustering structure. In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'99), Philadelphia, PA, USA*, 1999.

[Aha92] D.W. Aha. Tolerating noisy, irrelevant and novel attributes in in instance-based learning algorithms. *Int. Jurnal of Man-Machine Studies*, 36:267–287, 1992.

[AHCG00] S. Aksoy, R. M. Haralick, F. A. Cheikh, and M. Gabbouj. A weighted distance approach to relevance feedback. In *Proc. of the 14th Int. Conf. on Pattern Recognition (ICPR), Barcelona, Spain*, 2000.

[AKKS99] M. Ankerst, G. Kastenmüller, H.-P. Kriegel, and T. Seidl. 3d shape histograms for similarity search and classification in spatial databases. In *Proc. 6th Int. Symposium on Large Spatial Databases (SSD'99), Hong Kong, China*, volume 1651, pages 207–226, 1999.

[AKPS05] E. Achtert, H.-P. Kriegel, A. Pryakhin, and M. Schubert. Hierarchical density-based clustering for multi-represented objects. In *Workshop on Mining Complex Data (MCD'05), ICDM'05, Houston, TX*, 2005.

[AKPS06] E. Achtert, H.-P. Kriegel, A. Pryakhin, and M. Schubert. Clustering multi-represented objects using combination trees. In *Proc. of 10th Pacific-Asia Conf. on Advances in Knowledge Discovery and Data Mining (PAKDD'06), Singapore, 2006*, volume 3918, pages 174–178, 2006.

[ALSS95] R. Agrawal, K.-I. Lin, H.S. Sawhney, and K. Shim. Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In *Proc. 21st Int. Conf. on Very Large Data Bases (VLDB'95), Zurich, Switzerland*, pages 490–501, 1995.

[AY99] Y. A. Aslandogan and C. T. Yu. Techniques and systems for image and video retrieval. *IEEE Transactions on Knowledge and Data Engineering*, 11(1):56–63, 1999.

[BBA$^+$03] B. Boeckmann, A. Bairoch, R. Apweiler, M.-C. Blatter, A. Estreicher, E. Gasteiger, M.J. Martin, K. Michoud, C. O'Donovan, I. Phan, S. Pilbout, and M. Schneider. The swiss-prot protein knowledgebase and its supplement trembl in 2003. *Nucleic Acid Research*, 31:365–370, 2003.

[BBJ$^+$00] S. Berchtold, C. Böhm, H.V. Jagadish, H.-P. Kriegel, and J. Sander. Independent quantization: An index compression technique for high-dimensional spaces. In *Int. Conf. on Data Engineering, ICDE 2000.*, 2000.

[BBK01] C. Böhm, S. Berchthold, and D. Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Computing Surveys*, 3(33), 2001.

[BGK$^+$07] C. Böhm, M. Gruber, P. Kunath, A. Pryakhin, and M. Schubert. Prover: Probabilistic video retrieval using the gauss-tree (demonstration). to appear. In *Proc. 23rd Int. Conf. on Data Engineering (ICDE'07), Istanbul, Turkey*, 2007.

[BKK96] S. Berchtold, D. A. Keim, and H.-P. Kriegel. The X-Tree: An index structure for high-dimensional data. In *Proc. 22nd Int. Conf. on Very Large Data Bases (VLDB'96), Mumbai (Bombay), India*, pages 28–39, 1996.

[BKK$^+$03] S. Brecheisen, H.-P. Kriegel, P. Kröger, M. Pfeifle, and M. Schubert. Using sets of feature vectors for similarity search on voxelized CAD objects. In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'03), San Diego, CA, USA*, pages 587–598, 2003.

[BKK$^+$06] S. Brecheisen, H.-P. Kriegel, P. Kunath, A. Pryakhin, and F. Vorberger. Muscle: Music classification engine with user feedback (demonstration). In *Proc. of 10th Int. Conf. on Extending Database Technology (EDBT'06), Munich, Germany*, pages 1164–1167, 2006.

[BKKP04] S. Brecheisen, H.-P. Kriegel, P. Kröger, and M. Pfeifle. Visually mining through cluster hierarchies. In *Proc. of the 4th SIAM Int. Conf. on Data Mining (SDM'04), Lake Buena Vista, Florida, USA*, pages 400–412, 2004.

[BKKP06] S. Brecheisen, H.-P. Kriegel, P. Kunath, and A. Pryakhin. Hierarchical genre classification for large music collections. In *Proc. of 7th IEEE Int. Conf. on Multimedia and Expo (ICME'06), Toronto, Ontario, Canada*, pages 1385–1388, 2006.

[BKS$^+$04] B. Bustos, D. A. Keim, D. Saupe, T. Schreck, and D. V. Vranic. Using entropy impurity for improved 3d object similarity search. In *Proc. of 5th IEEE Int. Conf. on Multimedia and Expo (ICME'04), Taipei, Taiwan*, pages 1303–1306, 2004.

[BKS$^+$05a] B. Bustos, D. A. Keim, D. Saupe, T. Schreck, and D. V. Vranic. Feature-based similarity search in 3d object databases. *ACM Comput. Surv.*, 37(4):345–387, 2005.

[BKS05b] B. Bustos, D. A. Keim, and T. Schreck. A pivot-based index structure for combination of feature vectors. In *SAC '05: Proc. of the 2005 ACM Symposium on Applied computing*, pages 1180–1184. ACM Press, 2005.

[BKSS90] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-Tree: An efficient and robust access method for points and rectangles. In *SIGMOD Rec.*, pages 322–331, 1990.

[BM98] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *In Proc. of the eleventh annual Conf. on Computational learning theory (COLT'98)*, pages 92–100, 1998.

[BM99] H. Brighton and C. Mellish. On the consistency of information filters for lazy learning algorithms. In *Proc. of the 3rd European Conf. on Principles of Data Mining and Knowledge Discovery (PKDD'99)*, pages 283–288, 1999.

[BM02] H. Brighton and C. Mellish. *Advances in Instance Selection for Instance-Based Learning Algorithms*, volume 6. Kluwer Academic Publishers, 2002.

[BPS06a] C. Böhm, A. Pryakhin, and M. Schubert. The gauss-tree: Efficient object identification in databases of probabilistic feature vectors. In *Proc. 22nd Int. Conf. on Data Engineering (ICDE'06), Atlanta, GA, USA*, page 9, 2006.

[BPS06b] C. Böhm, A. Pryakhin, and M. Schubert. Probabilistic ranking queries on gaussians. In *Proc. of the 18th Int. Conf. on Scientific and Statistical Database Management (SSDBM'06)*, pages 169–178, 2006.

[BS04] S. Bickel and T. Scheffer. Multi-View Clustering. In *Proc. 4th IEEE Int. Conf. on Data Mining (ICDM'04), Brighton, UK*, 2004.

[Cam97] J. P. Campbell. Speaker recognition: A tutorial. Proc. of the IEEE, Vol. 85, No. 9, 1997.

[CH67] T.M. Cover and P.E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on information Theory*, IT-13:21–27, 1967.

[CKG02] S.-C. Chen, R.L. Kashyap, and A. Ghafoor. *Semantic Models for Multimedia Database Searching and Browsing.* Kluwer Academic Publishers, 2002.

[CKP03] R. Cheng, D.V. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'03), San Diego, CA, USA*, pages 551–562, 2003.

[CLC98] T. S. Chua, W. C. Low, and C. X. Chu. Relevance feedback techniques for color-based image retrieval. In *Proc. of the 4th IEEE Int. MultiMedia Modelling Conf. (MMM'98), Washington, D.C., USA*, 1998.

[Con00] The Gene Ontology Consortium. Gene ontology: Tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.

[CPZ97] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proc. of the 23rd Int. Conf. on Very Large Data Bases*, pages 426 – 435. Morgan Kaufmann, San Francisco, CA, USA, 1997.

[CSL99a] H. S. Chang, S. Sull, and S. U. Lee. Efficient video indexing scheme for content-based retrieval. In *IEEE Transactions on Circuits and Systems for Video Technology*, volume 9, pages 1269–1279, 1999.

[CSL99b] H. S. Chang, S. Sull, and S. U. Lee. Efficient video indexing scheme for content-based retrieval. In *IEEE Transactions on Circuits and Systems for Video Technology*, volume 9, pages 1269–1279, 1999.

[CV95] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[CVK04] C. H. L. Costa, J. D. Jr. Valle, and A. L. Koerich. Automatic classification of audio data. *IEEE Transactions on Systems, Man, and Cybernetics*, 3(6), 2004.

[CWS95] R. Chellappa, C. Wilson, and S. Sirohey. "Human and machine recognition of faces: A survey". Proc. IEEE, 83(5):705–740, 1995.

[CXP$^+$04] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, and J.S. Vitter. Efficient indexing methods for probabilistic threshold queries over uncertain data. In *Proc. 30th Int. Conf. on Very Large Data Bases (VLDB'04), Toronto, Cananda*, pages 876–887, 2004.

[CZ02a] S.S. Cheung and A. Zakhor. Efficient video similarity measurement with video signature. In *IEEE International Conference on Image Processing (ICIP 02)*, volume 1, pages 621–624, 2002.

[CZ02b] S.S. Cheung and A. Zakhor. Efficient video similarity measurement with video signature. In *Proc. of the IEEE Int. Conf. on Image Processing (ICIP'02), Rochester, New York, USA*, volume 1, pages 621–624, 2002.

[DD05] P. D. Dobson and A. J. Doig. Predicting enzyme class from protein structure without alignments. *J. Mol. Biol.*, 4(345):187–199, 2005.

[Deb04] S. Deb. *Multimedia Systems and Content-Based Retrieval.* Idea Group Publishing, 2004.

[Deb05] S. Deb. *Video Data Management and Information Retrieval.* Idea Group Publishing, 2005.

[DK02] M. Deshpande and G. Karypis. Evaluation of techniques for classifying biological sequences. In *Proc. of 6th Pacific-Asia Conf. on Advances in Knowledge Discovery and Data Mining (PAKDD'02), Taipei, Taiwan, 2002*, pages 417–431, 2002.

[DLLP97a] T.G. Dietterich, R.H. Lathrop, and T. Lozano-Perez. "Solving the multiple instance problem with axis-parallel rectangles". *Artificial Intelligence*, 89:31–71, 1997.

[DLLP97b] T.G. Dietterich, R.H. Lathrop, and T. Lozano-Perez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89:31–71, 1997.

[DM00] I. S. Dhillon and D. S. Modha. A data-clustering algorithm on distributed memory multiprocessors. In *Revised Papers from Large-Scale Parallel Data Mining, Workshop on Large-Scale Parallel KDD Systems, SIGKDD*, pages 245–260, 2000.

[DS05] V. R. De Sa. Spectral Clustering with two Views. In *Proc. ICML Workshop*, 2005.

[Dui02] R. Duin. The combining classifier: To train or not to train? In *Proc. 16th Int. Conf. on Pattern Recognition (ICPR'02), Quebec City, Canada)*, pages 765–770, 2002.

[DYM$^+$05] X. Dai, M. L. Yiu, N. Mamoulis, Y. Tao, and M. Vaitis. "Probabilistic Spatial Queries on Existentially Uncertain Data". In *Pro. 9th Int. Symposium on Spatial and Temporal Databases (SSTD2005),Angra dos Reis, Brazil*, pages 400–417, 2005.

[EKS02] M. Ester, H.-P. Kriegel, and M. Schubert. Web site mining: A new way to spot competitors, customers and suppliers on the WWW. In *Proc. 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD'02), Edmonton, Canada*, 2002.

[EKSX96] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD'96), Portland, OR, USA*, 1996.

[EM97] T. Eiter and H. Mannila. Distance measures for point sets and their computation. *Acta Informatica*, 34(2):103–133, 1997.

[FBF$^+$94] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, and etall. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3:231–262, 1994.

[FI92] U. M. Fayyad and K. B. Irani. On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, 8:87–102, 1992.

[FPSS96] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. Knowledge discovery and data mining: Towards a unifying framework. In *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD'96), Portland, OR, USA*, pages 82–88, 1996.

[FRB98] U. Fayyad, C. Reina, and P. Bradley. Initialization of iterative refinement clustering algorithms. In *Proc. 4th Int. Conf. on Knowledge Discovery and Data Mining (KDD'98), New York, NY, USA*, pages 194–198, 1998.

[FRM94] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. "Fast Subsequence Matching in Time-Series Databases". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'94), Minneapolis, MN*, pages 419–429, 1994.

[FZ00] G. Forman and B. Zhang. Distributed data clustering can be efficient and exact. *SIGKDD Explor. Newsl.*, 2(2):34–38, 2000.

[Gat72] G.W. Gates. The reduced nearest neighbour rule. *IEEE Transactions on Information Theory*, 18(3):431–433, 1972.

[GFKS02a] T. Gärtner, P. A. Flach, A. Kowalczyk, and A. Smola. Multi-instance kernels. In *Proc. 19th Int. Conf. on Machine Learning (ICML'02), Sydney, Australia*, pages 179–186, 2002.

[GFKS02b] T. Gärtner, P.A. Flach, A. Kowalczyk, and A. Smola. "Multi-Instance Kernels". In *Proc. 19th Int. Conf. on Machine Learning (ICML'02), Sydney, Australia*, pages 179–186, 2002.

[GGM02] H. Greenspan, J. Goldberger, and A. Mayer. A probabilistic framework for spatio-temporal video representation & indexing. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part IV*, pages 461–475. Springer-Verlag, 2002.

[Gut84] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 47–57, 1984.

[GW99] B. Ganter and R. Wille. *Formal Concept Analysis. Mathematical Foundations.* Springer, 1999.

[HB01] A. Hampapur and R. M. Bolle. Comparison of distance measures for video copy detection. In *IEEE International Conference on Multimedia and Expo (ICME'01)*, page 188, 2001.

[HBV01] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. "On Clustering Validation Techniques". *Journal of Intelligent Information Systems*, 2/3(17):107–145, 2001.

[HK06] J. Han and M. Kamber. *Data Mining: Concepts and Techniques. 2nd Ed.* Academic Press, 2006.

[HS95] G.I. Hjaltason and H. Samet. "Ranking in Spatial Databases". In *Proc. 4th Int. Symposium on Large Spatial Databases, SSD'95, Portland, USA*, volume 951, pages 83–95, 1995.

[HSD73] R. M. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *Proc. IEEE SMC*, 1973.

[HSW$^+$95] J. Hafner, H.S. Sawhney, Equitz W., M. Flickner, and W. Niblack. Efficient color histogram indexing for quadratic form distance functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 17 (7):729–736, 1995.

[IBW$^+$04] T. Ianeva, L. Boldareva, T. Westerveld, R. Cornacchia, A. de Vries, and D. Hiemstra. Probabilistic approaches to video retrieval. In *TREKVID*, 2004.

[IL00] G. Iyengar and A. B. Lippman. Distributional clustering for content-based retrieval of images and videos. In *Proc. Int. Conf. Image Processing*, pages 81–84, 2000.

[JK99] E. Johnson and H. Kargupta. Hierarchical clustering from distributed, heterogeneous data. In M. Zaki and C. Ho, editors, *Large-Scale Parallel KDD Systems*, volume 1759. Springer Verlag, 1999.

[JKP04] E. Januzaj, H.-P. Kriegel, and M. Pfeifle. Scalable density-based distributed clustering. In *Proc. Europ. Conf. PKDD*, 2004.

[KBD01] L.I. Kuncheva, J.C. Bezdek, and R.P.W. Duin. Decision templates for multiple classifier fusion: an experimental comparison. *Pattern Recognition*, 34:299–314, 2001.

[KDTM06] Y. Kanellopoulos, T. Dimopulos, C. Tjortjis, and C. Makris. Mining source code elements for comprehending object-oriented systems and evaluating their maintainability. *SIGKDD Explorations, 8(1):33–40*, 2006.

[KHDM98] J. Kittler, M. Hatef, R.P.W. Duin, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

[KKKP06] H.-P. Kriegel, P. Kröger, P. Kunath, and A. Pryakhin. Effective similarity search in multimedia databases using multiple representations. In *Proc. of the 12th IEEE Int. MultiMedia Modelling Conf. (MMM'06), Beijing, China*, pages 389–393, 2006.

[KKPS04a] K. Kailing, H.-P. Kriegel, A. Pryakhin, and M. Schubert. Clustering multi-represented objects with noise. In *Proceedings of Advances in Knowledge Discovery and Data Mining, 8th Pacific-Asia Conference, PAKDD 2004, Sydney, Australia.*, 2004.

[KKPS04b] H.-P. Kriegel, P. Kröger, A. Pryakhin, and M. Schubert. Using support vector machines for classifying large sets of multi-represented objects. In *Proc. of the 4th SIAM Int. Conf. on Data Mining (SDM'04), Lake Buena Vista, Florida, USA*. SIAM, 2004.

[KKPS05] H.-P. Kriegel, P. Kröger, A. Pryakhin, and M. Schubert. Effective and efficient distributed model-based clustering. In *Proc. 5th IEEE Int. Conf. on Data Mining (ICDM'05), Houston, TX, USA*, pages 258–265, 2005.

[KKSS04] K. Kailing, H.-P. Kriegel, S. Schönauer, and T. Seidl. Efficient similarity search for hierachical data in large databases. In *Proc. of 10th Int. Conf. on Extending Database Technology (EDBT'04), Heraklion, Greece*, pages 88–105, 2004.

[KPS05] H.-P. Kriegel, A. Pryakhin, and M. Schubert. Multi-represented knn-classification for large class sets. In *Proc. 10th Int. Conf. on Database Systems for Advanced Applications (DASFAA'05), Beijing, China*, volume 3453, pages 511–522, 2005.

[KPS06] H.-P. Kriegel, A. Pryakhin, and M. Schubert. An em-approach for clustering multi-instance objects. In *Proc. of 10th Pacific-Asia Conf. on Advances in Knowledge Discovery and Data Mining (PAKDD'06), Singapore, 2006*, pages 139–148, 2006.

[KPSZ06] H.-P. Kriegel, A. Pryakhin, M. Schubert, and A. Zimek. Cosmic: Conceptually specified multi-instance clusters. to appear. In *Proc. 6nd IEEE Int. Conf. on Data Mining (ICDM'06), Hong Kong*, 2006.

[Lev44] K. Levenberg. A method for the solution of certain problems in least squares. *Quart. Appl. Math.*, 2:164–168, 1944.

[LJF94] K.-I. Lin, H. V. Jagadish, and C. Faloutsos. "The TV-Tree: An Index Structure for High-Dimensional Data". *VLDB Journal*, 3(4):517–542, 1994.

[LKBR06] K. Liu, H. Kargupta, K. Bhaduri, and J. Ryan. Distributed data mining bibliography, 2006.

[LYH06] C. Liu, X. Yan, and J. Han. Mining control flow abnormality for logic error isolation. In *Proc. of the 6th SIAM Int. Conf. on Data Mining (SDM'06), Bethesda, MD, USA*, 2006.

[LYY$^+$05] C. Liu, X. Yan, H. Yu, J. Han, and P.S. Yu. Mining behaviour graphs for "backtrace" of noncrashing bugs. In *Proc. of the 5th SIAM Int. Conf. on Data Mining (SDM'05), Newport Beach, CA, USA*, 2005.

[MG93] R. Mehrotra and J.E. Gary. Feature-based retrieval of similar shapes. In *Proc. 9th Int. Conf. on Data Engineering (ICDE'93), Vienna, Austria*, pages 108–115, 1993.

[MPV05] Y. Manalopoulos, A. Papadopoulos, and M. G. Vassilakopoulos. *Spatial Databases: Technologies, Techniques and Trends.* Idea Group Publishing, 2005.

[NBE$^+$93] W. Niblack, R. Barber, W. Equitz, M. Flickner, H. E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin. The qbic project: Querying images by content using color, texture, and shape. In *SPIE 1993 Int. Symposium on Electronic Imaging: Science and Technology Conference 1908, Storage and Retrieval for Image and Video Databases, San Jose, CA*, 1993.

[NH94] R. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proc. 20th Int. Conf. on Very Large Data Bases (VLDB'94), Santiago, Chile*, pages 144–155, 1994.

[NHBM98] D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. UCI repository of machine learning databases, 1998.

[NWH01a] M. Naphade, R. Wang, and T. Huang. Multimodal pattern matching for audio-visual query and retrieval. In *Proc. of the Storage and Retrieval for Media Databases (SPIE), San Jose, CA*, pages 188–195, 2001.

[NWH01b] M. Naphade, R. Wang, and T. Huang. Multimodal pattern matching for audio-visual query and retrieval. In *Proc. SPIE, Storage and Retrieval for Media databases*, volume 4315, pages 188–195, 2001.

[oI84] Federal Bureau of Investigation. The science of fingerprints: Classification and uses. Washington, D.C., U.S. Government Printing Office, 1984.

[OS03] C.S. Ong and A. Smalo. Machine learning with hyperkernels. In *Proc. 20th Int. Conf. on Machine Learning (ICML'03), Washington, DC, USA*, pages 576–583, 2003.

[PCST99] J. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin dags for multiclass classification. In *Proc. Advances in Neural Information Processing Systems*, pages 547–553, 1999.

[PJ99] D. Pfoser and C. S. Jensen. Capturing the uncertainty of moving-object representations. *Lecture Notes in Computer Science*, 1651, 1999.

[PK03] B.-H. Park and H. Kargupta. Distributed data mining: Algorithms, systems, and applications. In N. Ye, editor, *The Handbook of Data Mining.* Lawrence Erlbaum Associates Publishers, 2003.

[RB01] J. Ramon and M. Bruynooghe. A polynomial time computable metric between points sets. *Acta Informatica*, 37:765–780, 2001.

[RHM97] Y. Rui, T. S. Huang, and S. Mehrotra. Content-based image retrieval with relevance feedback in mars. In *Proc. of the IEEE Int. Conf. on Image Processing (ICIP'97), Santa Barbara, CA, USA*, 1997.

[RWLI75] G.L. Ritter, H.B. Woodruff, S.R. Lowry, and T.L. Isenhour. An algorithm for the selective nearest neighbor decision rule. *IEEE Transactions on Information Theory*, 21(6):665–669, 1975.

[Sal89] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison-Wesley, 1989.

[SC99] M. Sanderson and B. Croft. Deriving concept hierarchies from text. In *Proc. 22nd ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR'99),Berkley, CA, USA*, pages 206–213, 1999.

[SEKX98] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu. Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. *Data Min. Knowl. Discov.*, 2(2):169–194, 1998.

[SJL+03a] J. R. Smith, A. Jaimes, C-Y. Lin, M. Naphade, A. P. Natsev, and B. Tseng. Interactive search fusion methods for video database retrieval. In *Proc. of the IEEE Int. Conf. on Image Processing (ICIP'03), Barcelona, Spain*, 2003.

[SJL+03b] J.R. Smith, A. Jaimes, C.-Y. Lin, M. Naphade, A.P. Natsev, and B. Tseng. Interactive search fusion methods for video database retrieval. In *IEEE International Conference on Image Processing (ICIP 03)*, volume 1, pages 741–744, 2003.

[SK97] T. Seidl and H.-P. Kriegel. Efficient user-adaptable similarity search in large multimedia databases. In *Proc. 23rd Int. Conf. on Very Large Data Bases (VLDB'97), Athens, Greece*, pages 506–515, 1997.

[Smy96] P. Smyth. Clustering using monte carlo cross-validation. In *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD'96), Portland, OR, USA*, pages 126–133, 1996.

[SN05] U. Srinivasan and N. Nepal. *Managing Multimedia Semantics.* IRM Press, 2005.

[SQL+03] J. Sander, X. Qin, Z. Lu, N. Niu, and A. Kovarsky. Automatic extraction of clusters from hierarchical clustering representations. In *Proc. of 7th Pacific-Asia Conf. on Advances in Knowledge Discovery and Data Mining (PAKDD'03), Seoul, Korea, 2003*, pages 75–87, 2003.

[SS00] M. Sayal and P. Scheuermann. A distributed algorithm for web-based access patterns. In *Proc. KDD-WS on Distributed and Parallel Knowledge Discovery*, 2000.

[TC02] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

[TCX$^+$05] Y. Tao, R. Cheng, X. Xiao, W.K. Ngai, B. Kao, and S. Prabhakar. "Indexing Multi-Dimensional Uncertain Data with Arbitrary Probability Density Functions". In *Proc. 31th Int. Conf. on Very Large Data Bases (VLDB'05), Trondheim, Norway*, pages 922–933, 2005.

[TK00] T. Tolonen and M. Karjalainen. A computationally efficient multipitch analysis model. *IEEE Transactions on Speech and Audio Processing*, 8(6), 2000.

[TKR99] Y.-P. Tan, S.R. Kulkarni, and P.J. Ramadge. A framework for measuring video similarity and its application to video query by example. In *IEEE International Conference on Image Processing (ICIP'99)*, volume 2, pages 106–110, 1999.

[TWZC02] G. Trajcevski, O. Wolfson, F. Zhang, and S. Chamberlain. The geometry of uncertainty in moving objects databases. In *Proc. of 10th Int. Conf. on Extending Database Technology (EDBT'02), Prague, Czech Republic*, pages 233–250, 2002.

[VM02] G. Valentini and F. Masulli. Ensembles of learning machines. In *Proc. of the 13th Italian Workshop on Neural Nets-Revised Papers (WIRN VIETRI 2002)*, pages 3–22, 2002.

[WFP03] N. Weidmann, E. Frank, and B. Pfahringer. A two-level learning method for generalized multi-instance problems. In *Proc. ECML 2003, Cavtat-Dubrovnik, Croatia, 2003*, pages 468–479, 2003.

[WLCS04] Y. Wu, C.-Y. Lin, E. Chang, and J. R. Smith. Multimodal information fusion for video concept detection. In *Proc. of the IEEE Int. Conf. on Image Processing (ICIP'04), Singapore*, 2004.

[WM97] H.R. Wilson and T.R. Martinez. Instance pruning techniques. In *Proc. 14th Int. Conf. on Machine Learning*, pages 403–411. Morgan Kaufmann Publishers, 1997.

[WM00] H.R. Wilson and T.R. Martinez. *Machine Learning, 38-3. Reduction Techniques for Instance-Based Learning Algorithms.* Kluwer Academic Publishers, Boston., 2000.

[WMSW01] J. T. L. Wang, Q. Ma, D. Shasha, and C. H. Wu. New techniques for extracting features from protein sequences. *IBM Systems Journal*, 40(2):426–441, 2001.

[WSB98] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proc. 24th Int. Conf. Very Large Data Bases, VLDB*, pages 194–205, 1998.

[WZ00] J. Wang and J.D. Zucker. Solving multiple-instance problem: A lazy learning approach. In *Proc. 17th Int. Conf. on Machine Learning (ICML'00), Stanford, CA, USA*, pages 1119–1125, 2000.

[WZC$^+$03] J. Wang, H. Zeng, Z. Chen, H. Lu, L. Tao, and W. Ma. ReCoM: Reinforcement clustering of multi-type interrelated data objects. In *Proc. SIGIR*, 2003.

[XJK03] X. Xu, J. Jäger, and H.-P. Kriegel. A fast parallel clustering algorithm for large spatial databases. *Data Mining and Knowledge Discovery, an International Journal*, 3(3):263–290, 2003.

[Yar95] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proc. of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196, 1995.

[ZCM02] H. Zeng, Z. Chen, and W. Ma. A unified framework for clustering heterogeneous web objects. In *Proc. 3rd WISE 2002, Singapore*, pages 161–172. IEEE Computer Society, 2002.

[ZCPR03] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Comput. Surv.*, 35(4):399–458, 2003.

[Zha03] T. Zhang. Semi-automatic approach for music classification. In *Proc. SPIE Conf. on Internet Multimedia Management Systems*, 2003.

[Zho04] Z.-H. Zhou. Multi-instance learning: A survey. Technical report, AI Lab, Computer Science and Technology Department, Nanjing University, Nanjing, China, 2004.

[Zhu05] O. V. Zhuravskii. *Error correction in an electromechanical angle sensor for an analog-to-digital conversion*, volume 34. Springer New York, 2005.

[ZRHM98] Y. Zhuang, Y. Rui, T. S. Huang, and S. Mehrotra. Adaptive key frame extraction using unsupervised clustering. In *Proc. of the IEEE Int. Conf. on Image Processing (ICIP'98), Chicago, IL, ISA*, 1998.